AFRL-HE-WP-TR-2006-0122

Perceptual Cognition in the Distributed Cognition (DCOG) Framework: A Study of Dual Coding and Temporal Factors in a Knowledge-Based Memory System

Robert G. Eggleston

Cognitive Systems Branch
Wright-Patterson AFB OH 45433

Katherine L. McCreight

N-Space Analysis
306 Winding Trail
Xenia, OH 45385

August 2006

Final Report for February 2001 to August 2006

Air Force Research Laboratory
Human Effectiveness Directorate
Warfighter Interface Division
Cognitive Systems Branch
WPAFB OH 45433-7604

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2006-0122

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

FOR THE DIRECTOR

//SIGNED//

DANIEL G. GODDARD
Chief, Warfighter Interface Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* August 2006	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* February 2001 - August 2006

4. TITLE AND SUBTITLE Perceptual Cognition in the Distributed Cognition (DCOG) Framework: A Study of Dual Coding and Temporal Factors in a Knowledge-Based Memory System	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 61102F
6. AUTHOR(S) [1]Robert G. Eggleston, [2]Katherine L. McCreight	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER 2313HC1A
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) [2]N-Space Analysis, 306 Winding Trail, Xenia, OH 45385	**8. PERFORMING ORGANIZATION REPORT NUMBER** TR NS 2001-3
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) [1]Air Force Materiel Command, Air Force Research Laboratory, Human Effectiveness Directorate, Warfighter Interface Division, Cognitive Systems Branch, Wright-Patterson AFB OH 45433-7604	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL-HECS
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-HE-WP-TR-2006-0122

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited. Cleared by AFRL/WS-06-2767 on 29 November 2006.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Distributed Cognition (DCOG) is a framework for the design and construction of computational models of cognitive behavior. The framework is based on a state-change theory of mind that emphasizes the interplay of recognitional and reasoning methods in the cognitive activities of acquiring and using knowledge. It attempts to provide a single coherent account of knowing, thinking, and learning forms of cognitive behavior. A key feature of the DCOG framework is the incorporation of an associative-based memory model that includes dual coding of conceptual knowledge. This produces a recognition-based system that is less brittle and more cognitively adaptable than other known modeling approaches. The DCOG framework is implemented as an agent-based system with a distributed control structure over 'mind domains,' each presumed to include an associative memory model that provides robust, flexible and adaptive knowledge formation and its use in work performance. This study explores the use of the dual coding of conceptual knowledge as a central component of the knowing process in perceptual cognition. It examines how a dual-coding memory system can model percept formation, and associated perceptual phenomena. More specifically, it focuses on the interaction of temporal factors with dual coding to account for perceptual 'knowing'.

15. SUBJECT TERMS Distributed Cognition (DCOG), Human Behavioral Representation, Cognitive Model, Perceptual Cognition Model, Human Performance Model, Visual Performance Modeling

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Robert G. Eggleston
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	SAR	250	**19b. TELEPHONE NUMBER** *(include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

THIS PAGE LEFT INTENTIONALLY BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE LEFT INTENTIONALLY BLANK

ABSTRACT

Distributed Cognition (DCOG) is a framework for the design and construction of computational models of cognitive behavior. The framework is based on a state-change theory of mind that emphasizes the interplay of recognitional and reasoning methods in the cognitive activities of acquiring and using knowledge. It attempts to provide a single coherent account of knowing, thinking, and learning forms of cognitive behavior (Eggleston, McCreight and Young, 2000, 2001, 2005).

A key feature of the DCOG framework is the incorporation of an associative-based memory model that includes dual coding of conceptual knowledge. This produces a recognition-based system that is less brittle and more cognitively adaptable than other known modeling approaches. The DCOG framework is implemented as an agent-based system with a distributed control structure over 'mind domains,' each presumed to include an associative memory model that provides robust, flexible and adaptive knowledge formation and its use in work performance.

This study explores the use of the dual coding of conceptual knowledge as a central component of the knowing process in perceptual cognition. It examines how a dual-coding memory system can model percept formation, and associated perceptual phenomena. More specifically, it focuses on the interaction of temporal factors with dual coding to account for perceptual 'knowing'. Three software programs demonstrate these factors in perceptual cognition. The first program shows how the knowledge based memory system handles the organization of static visual elements to form a unified gestalt percept. The second program shows how the same memory system structure can account for perceptual organization of dynamic (time-based) visual elements—i.e. apparent motion phenomena. It further illustrates how the memory system is able to account for perceptual hysteresis, a known property of human perception. The last program illustrates how the basic mind elements (called features) used in the memory system can be bootstrapped from more peripheral visual processing by a self-organizing method. Together, the three programs demonstrate how simple differences in temporal dynamics across dual coding elements can account for the formation of a holistic perception. It extends earlier DCOG research and suggests that dual-coding may be a foundational component for the localized memory systems of the human mind, one that supports flexible and adaptive cognitive behavior in a robust manner.

1.0 INTRODUCTION

Computational models of human behavior and performance continue to gain in importance for the development and use of tools to support Air Force personnel in war fighting operations. Software agents that emulate human cognitive abilities, for example, are embedded in many software applications that aid war fighter activities. Cognitive factors may be used in software to aid the war fighter in many ways: e.g. to determine what, when, and where task-critical alerts need to be provided, to interpret user input to a software tool or weapon system and act on that interpretation, to assist in information retrieval over a distributed network, to assist general aspects of knowledge management in planning, coordination, and execution, and to aid in the accomplishment of specific task details associated with cognitively demanding activities such as those found in a theater Air Operations Center. Further, additional areas where cognitive models are of value to the war fighter include war game simulations and simulations used in the acquisition and upgrading of weapon systems.

The degree of sophistication of cognitive-based automation ability needed in war fighter applications and simulation tools continues to grow. More cognitive ability of machine tools are needed to help the user handle higher volumes of available information and to handle more intricate and complex task scenarios against a smart, evolving adversary.

In previous reports we have introduced a new framework for the design and construction of computational models of cognitive behavior. This framework, known as

Distributed Cognition (DCOG), is based on a state-change theory of mind that emphasizes the interplay of recognitional methods and reasoning methods in the cognitive activities of acquiring and using knowledge in knowing, thinking, and learning processes (Eggleston, McCreight and Young, 2000, 2001, 2005). A key feature of the DCOG framework is the incorporation of an associative-based memory model that includes dual coding of conceptual knowledge. This produces a recognition-based system that is less brittle and more cognitively adaptable than other known modeling approaches. The DCOG framework is implemented as an agent-based system with a distributed control structure over 'mind domains,' each presumed to include an associative memory model that provides robust, flexible and adaptive learning and performance.

The purpose of this study was to investigate natural extensions of the DCOG framework, which has demonstrated successful cognitive performance in a complex air traffic management and leaning task (Eggleston, McCreight, and Young, 2005). For this study, we focused on the issue of temporal factors in the DCOG memory system. Our objective was to investigate how time factors in processing might interact with the dual-coding memory structure to influence knowledge recognition. As a first look into this issue, we chose to focus on perceptual cognition—i.e., the activation of a perceptual organization based on stored associational knowledge. Because some types of percepts depend on temporal factors for their existence, the perceptual domain should be a good place to start the examination of temporal interactions with the dual-coding memory system.

In order to examine a wider range of perceptual organization phenomena, we elected to minimize the amount of effort given to software code development. Our strategy was to produce software code that was consistent with properties of the DCOG framework, but to only encode those elements in software that were needed to test activation decay dynamics and self-organization behavior in the domain of perceptual cognition. Our tactic was to sample a range of perceptual cognitive phenomena to see if a common dual-coding memory structure with a common temporal design structure could account for behavior over a broad range of knowledge-based phenomena. Specifically, we examined regularization of percept formation for both formed-based and motion-based perceptual 'objects,' perceptual hysteresis, and the perception of multi-stable ambiguous figures.

We begin with a brief overview of the DCOG framework, emphasizing its memory system. Next, we briefly describe the dual-coding property of the memory system and introduce a basic temporal arrangement used with the dual-coding memory structure. This temporal pattern is then examined separately for each of the four perceptual phenomena investigated. The report closes with a general discussion of the study results in relation to the DCOG modeling framework.

2.0. DCOG FRAMEWORK OVERVIEW

DCOG is a framework for modeling human-like cognitive behavior (i.e., knowing, thinking, and learning). It is an attempt to provide a unified architecture for knowledge-based behavior and performance including the ability to handle methods for

(1) implicit (emergent) knowing, (2) multiple forms of deliberate reasoning, and for (3) real-time knowledge-level learning during task performance. It is intended to be a general architecture of mind that can be used as the computational engine to generate human-like behavior for use in software agents and other forms of machine automation.

The framework is based on four premises. We assume that the control of mental operations is distributed over a set of mind domains and mechanisms. There is no central control functionality. We believe there are different forms of knowledge-level cognition and that cognition pervades the behavior of highly intelligent life forms. In broad terms, we assert there are four principal domains in which different forms of cognitions can be evidenced: perceptual, symbolic, emotional, and motor domains.[1] Each of these domains has unique constraints and abilities that both focuses and limits cognitive behavior and thereby account for the existence of there unique abilities. In spite of these differences, we assume that there is a common core of structural and operational elements across domains that provide the central machinery for knowledge-level influences on complex human behavior. We further assume that in general individually simple structures and mechanisms are netted together to provide high-level cognitive abilities. These assumptions are consistent with an evolutionary development view of mind.

Cognitive abilities require the use of memory. The structure and operations of a core memory system, therefore, is a major component of each specialized domain of the mind system. The core memory system for DCOG is recognition based and employs simple structural and processing mechanisms.

[1] A fifth domain dealing with sensory transduction is also assumed. Information in this domain is taken to be most directly carried by varying energy patterns. The realm of cognition deals with subsequent processing, presumably involving knowledge element primitives formed from sensory domain processing. The focus of DCOG modeling begins with these basic knowledge elements.

Memory elements are both developed from and make use of an activation-association processing framework. A network structure of nodes represents the basic-level knowledge primitives over which activation and association operate. Activation is a temporal energy state that turns on selected nodes. Association is a persistent connection link formed between nodes to create a network that can code types and levels of meaning. The full network structure is termed the **backcloth**, following the Q-analytic framework developed by Atkin (1974). In the framework of Q-analysis, each node might represent a person, with the associations representing relationships between people. Alternatively, each node might represent a personal quality, with a cluster of connected nodes representing a particular person. Following Young (1998), we have adapted this system so that a cluster of connected nodes represents a mental concept.
Each individual node represents a feature that may be perceived, such as color, shape, etc. When a feature is perceived, the corresponding node is activated. Each node has a **firing threshold**, so that repeated input may be required for the node to become active. When a node fires becomes active, it fires, spreading activation energy to other nodes, in proportion to the connection strength of the links between the nodes. Such **spreading activation** may in turn cause other nodes to fire. When two nodes are **co-active**, the connection between them is strengthened by a process of **association**. Over time, the activation levels **decay**, so that a node representing an input which is not reinforced by either spreading activation or continued perceptual activation will drop below firing threshold and become inactive. Activation is thus a temporary phenomenon, which leaves traces in the form of associations between nodes.

When an object is perceived, various features are perceived. These features activate the corresponding nodes. Some nodes go over threshold and fire. Connection strengths between the active nodes are strengthened. When a similar object is subsequently perceived, many of the same nodes become active, further strengthening connections among the typical features of this type of object, creating the basis for perceptual cognition. Although the exemplars of the concept have individual differences, over time the associational memory builds up a core of strongly connected nodes which represent the concept in a manner loosely connected to contextual aspects of the initiating context. We term this **emergent concept formation**.

Because concepts are built up from fuzzy sets of partially overlapping exemplars, they are free to evolve. Such **concept drift** allows the concept to continue to be the most useful representation for current circumstances, one that can be readily embedded and related to the current context.

We define **concept recognition** as the synchronous activation of the core features of a concept. (That is, all the nodes in the concept's core are activated above threshold and firing in synchrony.) Concept recognition may be the result of direct activation from perceptual input, or it may depend on both direct and indirect (spreading) activation. Context factors, expressed as feature nodes can indirectly influence concept recognition. Initial activation of a subset of the core features may suffice for recognition, since activation will then spread among the strongly connected nodes.

Concept recognition may stimulate the creation of feature nodes that become part of the backcloth. It may also activate simple attention mechanisms, which in turn may change the required threshold level used to define the active portion of the node network and thus bring new concept-based knowledge into awareness.[2] Recognition may lead directly to action behavior executed through the motor domain or additional reasoning may be involved.

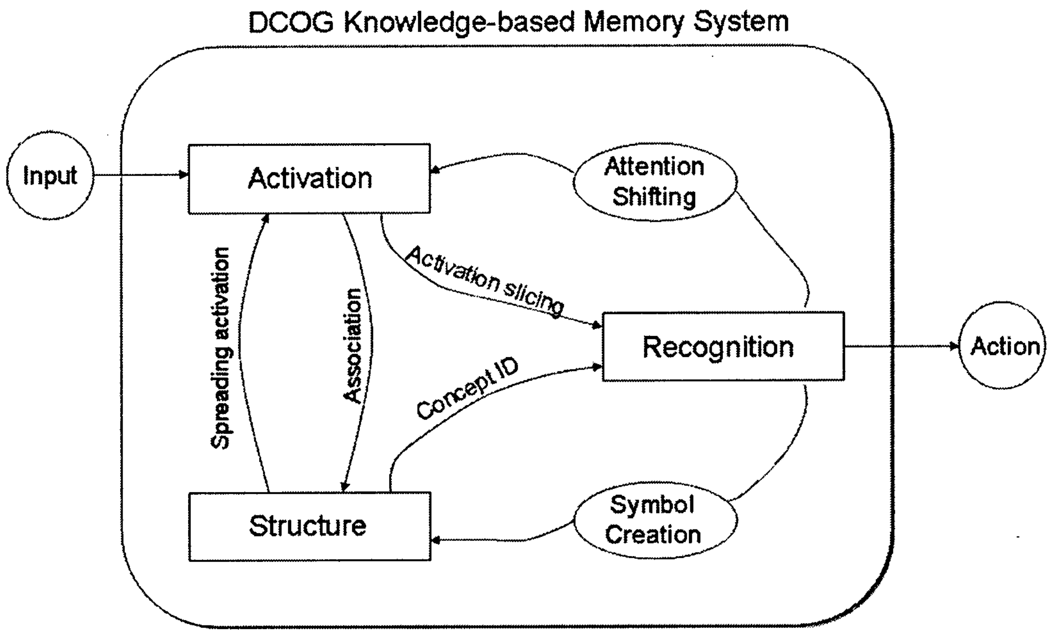


Figure 1. Simplified schematic of the DCOG memory model.

A simplified schematic of the DCOG memory system, presented in Figure 1, summarizes the operation of the DCOG memory system. Activation turns on associatively linked nodes, which, in turn, yields spreading of activation over more links in the node structure. There is a low level attentional mechanism that slices or filters the active node network to

[2] While attention is an important construct in the DCOG framework, there is no monolithic, centralized attention function, per se. Attention is achieved through simple mechanisms like threshold level selection.

select stored knowledge (concepts). Stored concepts are compared with the input activation to achieve knowledge-based recognition. Once recognition has occurred, concept memory is updated, which may result in a change in concept structure, and a change in attention can be induced to support the stringing of concepts into a more complex 'thought.'

We can see elemental forms of knowing, thinking, and learning in the DCOG memory model. Knowing is reflected in the emergent concept formation. Thinking is reflected in spreading activation and attention shifting. Learning is reflected in the changing association strengths within the node network. The interplay of these forms of cognitive abilities is addressed in Eggleston, McCreight, and Young (2005), which describes the behavior and performance of a DCOG model used as a software agent engaged in an air traffic control task that required real-time learning. A more detailed description of the basic DCOG memory model is provided in Eggleston and McCreight (2006).

3.0. DUAL CODING AND ACTIVATION DECAY

One of the main properties of the DCOG framework is a memory system that involves dual coding of concepts. In previous DCOG research, concepts were coded as concept nodes linked to feature nodes with various weights, as illustrated in Figure 2.

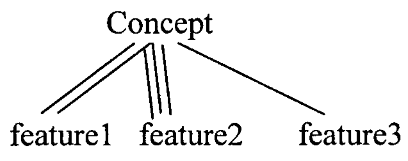


Figure 2. Dual-Coding Structure of the DCOG memory system.

This enables the DCOG system to model the recognition of fuzzy concepts. When a DCOG agent views a document title or key words, for example, a set of terms in it activates a given concept in memory when the sum of the links from those terms to the concept exceeds a certain proportion of the total term links for that concept. The concept may be activated by a document containing one strongly-linked term (i.e., a good access node.), or by a document containing several weakly-linked terms, including some of which that may be regarded as context items. This dual coding structure is a fundamental component of the DCOG knowledge-based memory model.

The current project shows how dual coding interacts with timing properties to support several perceptual phenomena, including object persistence, regularization of input, perceptual hysteresis, and perceptual behavior with ambiguous figures. These phenomena are demonstrated in two programs, Cycle.java and ApparentMotion.java, which both rely on dual coding structures.

A third program, Domain.java, introduces a linear mechanism which models the self-organization of firing nodes. Such a mechanism could be used to model input energy patterns into information-carrying feature nodes that serves as the basis for a concept-based (or knowledge based) memory system. Together, the dual coding structure and the self-organization algorithm suggest how perceptual inputs may be successively organized into objects, which can carry conceptual meaning.

We do not claim to provide a precise model of human perceptual processing. Although our model starts at the "feature" level, we are actually modeling a level which is already somewhat removed from the external environment. We assume, but do not model, some earlier regularization of the input that might be accomplished by more peripheral sensory processing mechanisms operating more directly in an energy domain.

Our model assumes a system of multiple types of mental domains (perceptual, symbolic, emotional, motor), each with its own memory nodes and decay rates. The sensory system that feeds the perceptual domain forms and decays data input quickly, while the perceptual system is more persistent, and the symbolically oriented cognitive system is the most persistent. In a similar manner, we assume activated features in the dual-coding memory structure of a domain decays more quickly than activated concepts (tightly coupled feature sets). We treat this simple temporal arrangement as a basic temporal structure of the memory system in each domain. We use this temporal schema, along with the basic dual-coding memory system structures and processes not as precise neuron models, but as guiding principles for the organization of a complex computational system.

3.1. Dual-Coding and Ambiguous Figures

This section describes the program, Cycle.java. In this program, we extend the dual coding system of previous DCOG programs, introducing an intermediate "object" level. The three-level structure is shown in Figure 3.

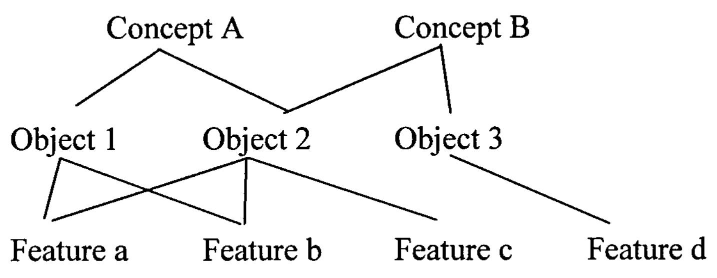


Figure 3. Dual Coding Structure.

This structure permits interactions between concepts and objects, and between objects and features. We also permit lateral connections, as for example, when a square object is made up of line and corner component objects, which are themselves composed of features. As before, the structure supports the recognition of a higher level entity based on partial, lower level input.

Features are represented as 3x3 bitmaps. Each section of the input is compared against possible features; when the number of matching bits exceeds a threshold, the feature detector receives activation. Currently, the input threshold is set to 0.8, so that the bitmap must match 8/9 bits to activate the feature detector. The feature detector accumulates activation both from the bitmap input, and from the higher-level objects which link to the feature. When the total activation exceeds a feature threshold, the feature fires, spreading activation to any linked objects. This spreading activation allows the system to recognize objects, based on their component features.

A sample of the feature bitmaps is indicated in Figure 4. We use compass directions to indicate the orientation of a corner or an intersection.

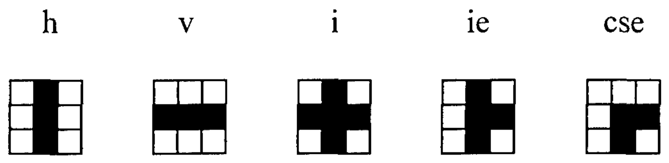


Figure 4. Some Feature BitGrids.

Objects are represented as nodes linking to certain component nodes. The components of an object may be features or other objects. Lateral object connections are used in the definition of a square, which exists at the object level, and is recognized according to its object components, as illustrated in Figure 5.

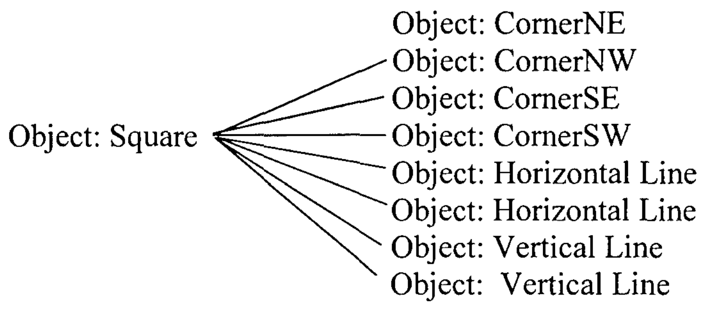


Figure 5. Lateral Associations.

This provides a lateral dual-coding of the square entity: a square is simultaneously the single object, square, and the combination of objects that includes four corners and four sides. Fuzzy recognition is possible here, as well; the square node may be activated, based on a subset of its component objects.[3]

Regularization of Input

The recognition of an object based on partial input provides the necessary structure to model regularization of input. Regularization of input refers to the perception of regularity not present in the physical input. Here, the system may recognize a square from input which is missing one side; subsequently, the system spreads activation from the square object to its component objects, and activates the missing element. This process is illustrated in Figures 6 through 9.

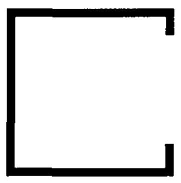


Figure 6. Input with Missing Line.

[3] In order to improve processing efficiency, the program does not analyze every combination of features and objects that could compose an object or a concept; instead, the program is aware of certain trigger features and objects that suggest the presence of an object or concept. Once these triggers are identified, the program then checks to see how many of the components for that entity are present. In recognizing a square object, for example, the program is sensitive to pairs of corners which are properly aligned to define a square. Once the program notices a potential square, based on a pair of corners, the program then checks for the presence of the four corners and four sides that would compose that square. This models the normal spreading activation from the component objects.

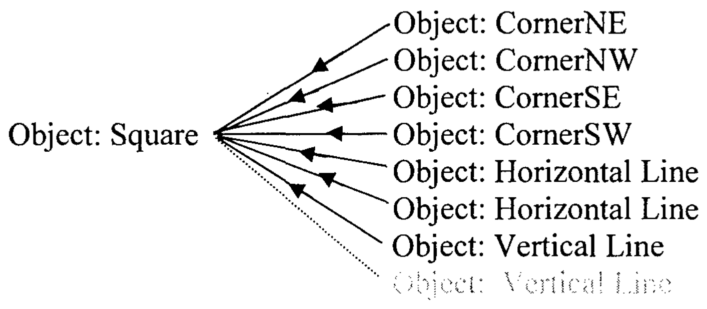


Figure 7. Active Objects Feed Activation to Square Object.

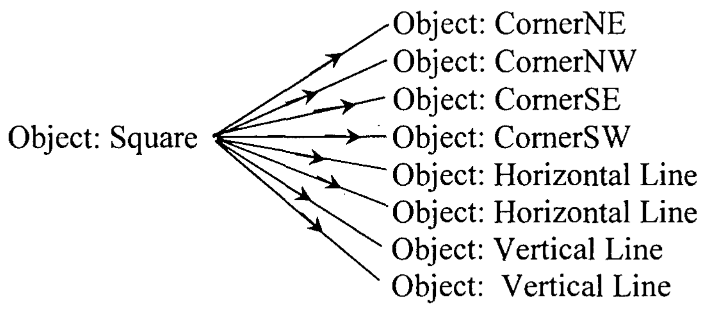


Figure 8. Square Object Feeds Activation to All Components.

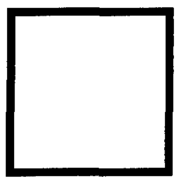


Figure 9. Perceived Input Includes Missing Line.

The links between the square object and its component objects represent our past experience or learning about the components of a square; based on this learning, the system "expects" the square to have all four sides. The newly activated side object, in turn, spreads activation down to its component features, so that the system perceives the features consistent with the missing line.

At time 4, the system recognizes the square object, based on the line and corner objects. The square subsequently spreads activation to its component objects. At time 10, the missing vertical line object becomes active.

Regularization of input is a form of context-dependent recognition (Eggleston and McCreight, 2006). Regularization of input helps the system recognize degraded or noisy input; in the example discussed above, the context of the three sides of a square allowed the system to perceive a segment that was totally absent in the input. It would be interesting to compare this system with human performance in other examples of context-dependent recognition, such as the differential perception of sketched facial features with and without a facial outline (Palmer 1975, cited in Rock 1983).

Decay Rates and Object Persistence

The dual-coding system includes different decay rates at each level. Features decay quickly, objects decay more slowly, and concepts have the slowest rate of decay. This means that constructs at a higher level may persist, even after the lower level constructs that triggered them have decayed. For example, if we show the system a horizontal line, followed by a blank screen, the perception of the horizontal line object lingers for a while after the physical input has disappeared. The system "remembers" that it recognized the line.[4]

In the current program, we have set the feature decay interval to 1, the object decay interval to 2, and the concept decay interval to 4. The program processes bitmap input at each unit of time, deriving and decaying features. The program does not derive

[4] Note that, if there is no decay built into the system, the persistent object would re-activate its component features, which would in turn re-activate the object. Without decay, the system would lock into a perception and be unable to "forget" it.

and decay objects until 2 units of time have passed; and concepts are updated only every 4 units of time. So, in the case of the disappearing line, we have the sequence of events shown in Table 1.

Table 1. Object Persistence.

Time	Bitmap	Features	Object Processed	Active Object
1	---	-,-,-		
2	---	-,-,-	---	---
3	---	-,-,-		--- (persists)
4	---	-,-,-	---	---
5				--- (persists)
6				
7				
8				

We see that the perception of the line object lingers into time 5, even though the input has changed to a blank screen. At time 6, object decay takes place, and the line is no longer perceived.[5, 6]

Attention-Based Resolution of Ambiguous Figures

There are some figures which two or more mutually exclusive interpretations. The Necker Cube is such a figure. Human subjects report seeing first one view of the cube, and then suddenly seeing an alternative view (see Figure 10).

[5] The output file shows that the system perceives corners at the ends of the line segment. This is because a terminal line bitmap differs from a corner bitmap by only one bit, and so is considered a match.
[6] It is important to recall that we are not modeling time-based decay associated with peripheral sensor-level mechanisms responsible for visual afterimages. Rather, we assume peripheral processing has occurred and our model then addresses the up stream temporal dynamics that may account for percept formation and consolidation processes.

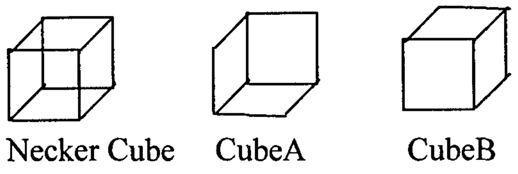


Figure 10. Necker Cube and Two Interpretations.

The perceptual system does not settle into a single interpretation, but continues to switch back and forth. This behavior has motivated the use of non-linear oscillators in the analysis of visual perception.

We do not attempt to model the range of oscillations in interpretation of the cube, but we do show that a simple linear mechanism can provide attention-based switching. Again, we rely on the dual coding mechanism.

The cube concepts in our model are coded both as concept nodes and as collections of objects, including a square, a central corner, and various horizontal, vertical, and diagonal lines. Most of these objects are common to both cube concepts, although they are found in different spatial relationships for the different concepts. The most distinct components are the central corners, shown in Figures 11 and 12.[7]

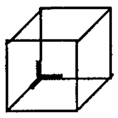


Figure 11. CornerNE3D.

[7] The trigger features which cause the system to check for a possible cube include the square, the central corner, and the opposing horizontal line.

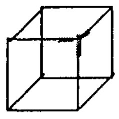


Figure 12. CornerSW3D.

These central corners appear three-dimensional, and, when taken as convex, correspond to the interpretation of the cube that places the corner on the near plane. The square object aligns with the central corner, in the near plane, while the independent horizontal and vertical line objects appear to occupy the far plane.

The system represents the central corners as both features and objects. Simple corner features are named according to the direction in which the corner opens, assuming that north is at the top of the screen. A corner shaped like the letter "L" is therefore represented by a feature named cne, corner opening to the northeast. The bitmap for a central corner includes one additional bit, indicating the start of the diagonal line. The features for these corners have names like cnedotsw, "a corner opening to the northeast, with an additional dot to the southwest." See Figure 13.

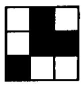


Figure 13. cnedotsw Feature Bitmap.

The objects corresponding to these corners represent a three-dimensional interpretation of the bitmap, and are named accordingly[8]: CornerNE3D, for example, is a corner opening to the northeast, with an associated diagonal which lends it a three-dimensional quality.

[8] For the Necker cube exemplar, this corner object could reflect the stereoscopic processing, based on differences in sensory input between two eyes, and supports the formation of a 3-D percept.

For human subjects, attention to the central corners may predispose perception of the corresponding cube.[9] We added a method to the system which provides selective attention to one of the central corners of the figure, and uses spreading activation to influence the selection of the cube interpretation.[10]

Attention to a particular object has the effect of an additional input at an intermediate time, between continued presentation of the entire figure. The components needed to trigger recognition of both cube concepts recur after the application of attention, as indicated in Figure 14.

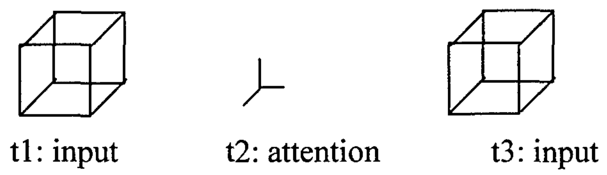


Figure 14. Sequence of Inputs and Attention.

The central corner object, CornerNE3D, is associated with the CubeA concept; the central corner object, CornerSW3D, is associated with the CubeB concept. These associations represent prior learning about the shapes associated with each perspective of the cube. When the system attends to a central corner, activation spreads along these associations, so that one interpretation of the cube achieves a higher level of activation

[9] It is not clear why the attended corner is interpreted as convex, and not concave. It may be that attending to one such corner both supports one interpretation of the cube, and inhibits the other interpretation, by interrupting the square which would lie on the near plane in the other interpretation. Such interpretive tensions may be addressed by mechanisms involved in managing stereopsis. For this project we do not address these more detailed aspects of stereoscopic percept formation and only model a positive influence of the central corner on the corresponding cube interpretation.

[10] There are several possible implementations of attention, including attention to a specific object in a specific location, attention to all objects of a given type, and attention to any object found in a specific location. The program code provides functions for each of these possibilities; we use a function which provides attention to a specific object in a specific location. Future work should consider which of these is the best representation of the human attention in this situation.

than the other. When concepts are processed, the activation levels are kept in a processing buffer, until conflicts can be identified and resolved. For example, suppose we attend to the central corner object, CornerSW3D, and thus concept CubeB comes to have a higher activation than CubeA. If concept CubeA has activation of 1.0, and concept CubeB has activation of 1.5, they both exceed the concept firing threshold. However, instead of allowing both to fire, the system identifies the conflict and allows only the concept with the highest activation to fire. In this example, CubeB is the single interpretation assigned to this input.

Given the varying decay rates, the process of switching from one interpretation to the other is delayed until the next concept-processing interval. Recall that objects are processed at intervals of 2 time units, while concepts are processed at intervals of 4 units. The sequence of events which occurs when shifting attention from one central corner to another is illustrated in Table 2. Here, we omit the input and features, since the input is held constant as the Necker Cube bitmap, and the features do not figure in the resolution of the conflict.

Table 2. Decay and Shifting Attention in the Necker Cube
s1 = top square, s2=bottom square, etc.=other objects (lines, corners).

Time	Processed Objects	Attended Object	Processed Concepts	Concept Activation	Perceived Concepts
1					
2	s1, s2, etc.				
3					
4	s1, s2, etc.		CubeA CubeB	CubeA=0.0 CubeB=0.0	CubeA CubeB
4.1		CornerNE3D		CubeA=0.5 CubeB=0.0	
5				CubeA=0.5 CubeB=0.0	CubeA CubeB
6	s1, s2, etc.			CubeA=1.0 CubeB=0.5	CubeA CubeB
7				CubeA=1.0 CubeB=0.5	CubeA CubeB
8	s1, s2, etc.		CubeA CubeB	CubeA=0.0 CubeB=0.0	CubeA
8.1		CornerSW3D		CubeA=0.0 CubeB=0.5	
9				CubeA=0.0 CubeB=0.5	CubeA
10	s1, s2, etc.			CubeA=0.5 CubeB=1.0	CubeA
11				CubeA=0.5 CubeB=1.0	CubeA
12	s1, s2, etc.		CubeA CubeB	CubeA=0.0 CubeB=0.0	 CubeB

No concepts are processed until time 4. Then, component objects provide activation to both CubeA and CubeB. Concept activation for each becomes 0.5 (not shown in the table), and is then reset to zero as each cube fires.

Our system differs from human perception here, in that both cubes are perceived when their activations are equal. This confused state does not occur in humans when observing the Necker cube. We suspect that human processing either invokes other resolution principles in this situation, or transitions out of this phase very quickly as the person scans the object, providing attention to various components.

After processing at time 4, we direct the program to attend to CornerNE3D.[11] This central corner object spreads activation to the CubeA concept, resulting in the higher level of activation of CubeA seen at time 5. However, the perceived concept list cannot change until concepts are processed again, at time 8. Continued input in the meantime increases activation of both concepts, so that, upon entering time 8, CubeA has an activation of 1.0 and CubeB has an activation of 0.5. The system resolves the conflict in favor of CubeA, which fires, and CubeA activation is reset to zero. The non-preferred cube concept is inhibited, so the activation of CubeB is also set to zero.

After time 8, we direct the program to attend to the other central corner, CornerSW3D, thus providing preferential activation to CubeB. The CubeA concept continues to be perceived until concepts are again processed at time 12, when CubeB becomes the perceived concept.

We use an activation buffer to represent some high-level perceptual reasoning about conflicting interpretations. This is inconsistent with the use of activation to

[11] In humans, this location shift in attention could be the result of normal micro saccadic eye movements, a change in locus of control from foveal to peripheral vision system, or due to a consciously directed saccadic eye movement.

represent low-level firing, spreading activation, and inhibition, in which a node which exceeds its activation threshold immediately fires. However, it seems clear that there is some kind of processing which corresponds to this conflict resolution, and that it is sensitive to comparative levels of activation. Future work may identify a representation of this processing buffer that is more consistent with our basic activation-association architectural style of data processing.

3.2. Apparent Motion

A second program, ApparentMotion.java, was created to examine apparent motion (Eggleston, 1986). The apparent motion program simulates a perceptual reaction to a sequence of dots which fall on the corners of an implied square. The sequence of dots may be interpreted as motion east and motion west, or as motion north and motion south (using north to indicate the top of the screen). The desired results are that the direction of motion is subject to a proximity constraint, but that this may be overruled by priming based on a history of activation within delimited prior time window. Priming can be effected by the apparent motion of a previous dot, or by the motion perceived on a previous trial.

The implementation for the apparent motion program is similar to the Necker cube program, in that it uses a dual-coding structure. Instead of feature-object-concept, however, we have feature-segment-path.[12] This modification reflects the change in nature of the class of percepts that are formed and recognized. The Necker cube is a spatially dominate percept in that all of its elements co-occur in time. Thus, features and

[12] The program also provides for feature-level motion nodes, representing initial detection of motion, but these do not play a role in the apparent motion simulations

objects are all available simultaneously. An apparent motion percept defines a different class of percept, one dominate by temporal factors. For this percept class, all features are not available simultaneously. Their time sequence induces the experienced perceptual geometry that results in seeing either a moving object or object morphing. We capture this type of percept formation by the feature-segment-path dual-coding structure of a perceptual memory system.

For our initial investigation of this aspect of perception the feature detection system was simplified, detecting a single feature, a dot, and the concept system was changed to provide motion concepts. The program perceives motion by keeping lists of the dots perceived at two successive input times. A succession of two dots activates a motion segment at the object level. The general direction of motion is recorded in a path. For north or south motion, the path also encodes the x-corridor. For east or west motion, the path encodes the y-corridor. For example, a path might encode motion south along the corridor, x=3.

When two segments have the same origin, they are considered to be in conflict. This expresses a presumed higher-level reasoning process, that avoids an interpretation in which the input as a single object splitting and traveling in two directions. The program contains a conflict resolution algorithm, so that when two co-originating motion segments are possible, the one with the highest activation is chosen. As with the Necker cube program, we must retain activation in a buffer, so that conflicts may be resolved before the segments fire.

The program first identifies the dots at the feature level. Given dots at two input times, the program detects motion segments. The segments receive activation according to the proximity of the endpoints. Next, the program identifies any active paths, and spreads activation from paths to motion segments that lie along those paths. The selection of e/w or n/s motion segments thus depends on both proximity and previously existing paths.

For most of the simulations, the bitgrids are presented in pairs, with the one named t1X presented at time 1, and the one named t2X presented at time 2. Together, the t1 and t2 inputs create an implied square. The t1 grid always includes the sw and ne points; the t2 grid always includes the nw and se points. The sw point in t1 is always at placed (7,11). The vertical height of the implied square is always 4 bits. The horizontal displacement is reflected in the name: For example, t1h2 indicates a horizontal displacement of 2 bits, while t1h3 indicates a horizontal displacement of 3 bits. In the following figures (Figures 15-20), gray bits represent input at time t1, while black bits represent subsequent input at time t2.

A single trial consists of a t1 bitgrid, followed by a t2 bitgrid. The t1 bitgrid presents the points corresponding to the sw and ne corners of the implied square, while the t2 bitgrid presents the points corresponding to the nw and se corners.

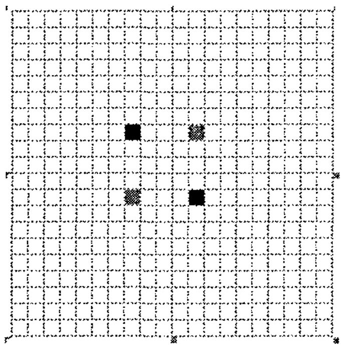


Figure 15. Basic Input Sequence for Apparent Motion: t1h4 (gray), t2h4 (black).

The input t1h4/t2h4 is used in the "balanced" simulation, since the horizontal displacement is equal to the vertical displacement. In principle, the balanced presentation can lead to either east/west apparent motion, or north/south apparent motion. There are some complications, which we discuss below.

The "tall" simulation uses the t1h3/t2h3 input, since the horizontal displacement (3) is less than the vertical displacement (4). The "wide" simulation uses the t1h5/t2h5 input, in which the horizontal displacement (5) is greater than the vertical displacement, (4).

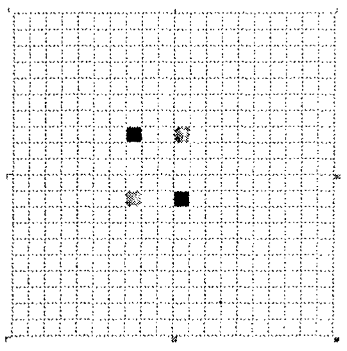


Figure 16. Tall Sequence: t1h3 (gray), t2h3 (black)

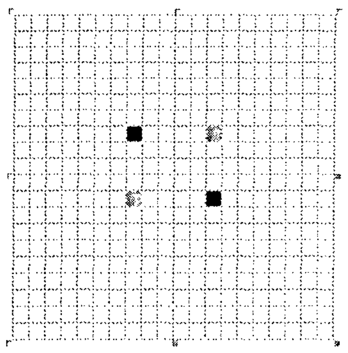


Figure 17. Wide Sequence: t1h5 (gray), t2h5 (black).

We express the proximity constraint as a calculation which provides segment activation in inverse proportion to the distance between the segment endpoints. When a single trial is presented, the program perceives the motion with the smallest displacement. When the vertical displacement is 4 and the horizontal displacement is 3, the program perceives east/west movement. When the vertical displacement is 4 and the horizontal displacement is 5, the program perceives north/south movement.

Now we turn to the implementation of priming. This depends on dual coding and differential decay rates. For this program, we implement differential decay by having the feature and segment elements cleared between presentations, while the path decay function reduces path activation by a small amount between trials. This allows path activation to persist across trials, providing priming for any motion consistent with a previously established path.

Priming effects are seen with the addition of dots at an earlier input time, and in the evaluation of input involving a sequence of trials with increasing or decreasing horizontal displacement.

For the horizontal priming and vertical priming simulations, a sequence of three bit grids is used. These consist of the t1h4/t2h4 bit grids, preceded by a bit grid in which two dots are placed either to the side or the top of the t1h4 dots, with a displacement equal to 4 bits (see diagrams, below). Here, blue bits are used to indicate the additional dots, presented at time t0.

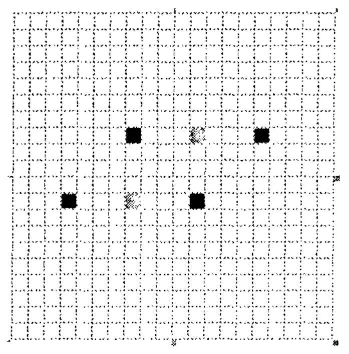


Figure 18. Horizontal Priming Sequence: t0h (blue), t1h4 (gray), t2h4 (black).

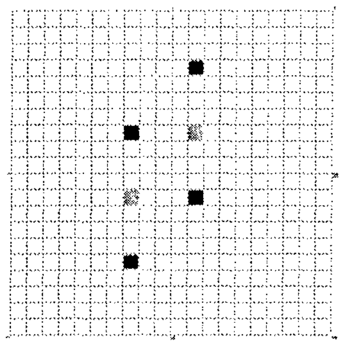


Figure 19. Vertical Priming Sequence: t0v (blue), t1h4 (gray), t2h4 (black).

Based on the t0-t1 time interval, the program activates motion segments and paths corresponding to that initial relationship. Path activation decays slowly, so the paths established at t1 are able to influence perception at time t2. Spreading activation from these paths ensures that the motion segments which continue the initial direction of motion are more highly activated, and hence perceived. When the initial dots are placed above and below the square, the system reports north/south motion throughout. When the initial dots are placed to the sides of the square, the system reports east/west motion throughout.

Next, we discuss the results of the simulations involving multiple trials with either increasing or decreasing horizontal displacement. The complete set of bit grids are given schematically in Table 3, below. For increasing horizontal displacement, the initial trial is perceived as east/west movement, based on the fact that the horizontal displacement is less than the vertical displacement. This east/west perception persists as horizontal displacement increases, even after the horizontal displacement exceeds the vertical displacement. Similarly, for decreasing horizontal displacement, an initial north/south perception persists, even after vertical displacement comes to exceed horizontal displacement.

This is similar to the object persistence we demonstrated previously, where a higher level node persists, even after the lower level nodes which activated it have decayed. Here, an initial motion segment leads to the activation of a higher level path element; this motion path persists even after the segments which activated the path have been removed. The presence of the path creates the effect of an expectation for continued movement in the same direction.

The motion perceptions for the increasing, decreasing, and independent trials are presented below (Table 3). A pair of bit grids is indicated by the notation hn, where the first bit grid is t1hn, and the second bit grid is t2hn, for some horizontal displacement value n. For the increasing sequence, the first trial is the leftmost one in the chart (bit grids h2), while for the decreasing simulation, the first trial is the rightmost one in the chart (bit grids h8).

Table 3. Perceived Motion.

Bit Set	h2	h3	h4	h5	h6	h7	h8
	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *
Single Trial	E/W	E/W	E/W	N/S	N/S	N/S	N/S
Increasing Trials	E/W	E/W	E/W	E/W	E/W	N/S	N/S
Decreasing Trials	E/W	N/S	N/S	N/S	N/S	N/S	N/S

For the single trials, proximity determines the direction of perceived motion. For the balanced input (h4), either north/south or east/west motion should be possible; in practice, the program chooses east/west motion (see discussion at the end of this section).

The increasing and decreasing trials exhibit priming across trials. Just as a horizontally placed dot primed the perception of horizontal motion in the balanced input condition (h4), the horizontal motion perceived in a previous trial (h3) primes perception of horizontal motion in the balanced input condition, and the vertical motion perceived in a previous trial (h5) primes perception of vertical motion in the balanced input condition (h4).[13] These priming effects continue, until the difference in vertical and horizontal displacement is sufficient to overcome them. The system thus exhibits hysteresis: The change in perception happens at a different point, depending on the direction of change of the horizontal displacement variable.

[13] For priming of north/south motion, we find a complication: Because the horizontal displacement is changing from trial to trial, the x-corridor for motion south is shifting. The x-corridor for motion north remains fixed at x=7 , since the initial lower dot is fixed at (7,11), but the x-corridor for motion south is changed by one bit in each successive trial. The program was adjusted to allow paths to prime movement in the original corridor, or in a corridor one bit away from the original. The question remains, how is this type of shift handled in human processing?

Finally, we discuss certain complications in the perception of balanced input (t1h4/t2h4). When the horizontal and vertical displacements are equal, proximity-based activations are balanced. If no priming interferes, there is no activation-based preference for one movement over another. If we let the program choose randomly, it could select, for example, north and west motion. The two dots from time t1 would converge on a single dot at time t2. This would leave no apparent source for the other dot at time t2; this dot would appear to simply blink on.

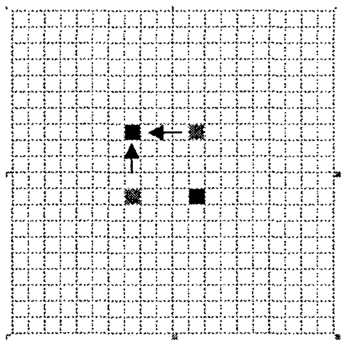


Figure 20. Merging Motion.

Human subjects do not exhibit this behavior, but always perceive matching motions, north and south, or east and west. We presume there is some higher-level processing which avoids interpretations in which objects blink on and off, and which also tends to avoid interpretations in which two objects move to a single location.[14] Future work might express these conditions, as well as the tendency to avoid a splitting interpretation, via some kind of constraint satisfaction. Alternatively, these constraints

[14] We are not suggesting that there is a specific process to handle object blinking or object splitting. Under suitable space-time configuration of features both types of percepts can be (formed) observed by humans. More research will be needed to determine if these aspects of perception can be handled in a compatible manner within our activation-association, dual-coding style of perceptual cognition. This may involve coordination among form, orientation, and motion processing pathways to yield perceptual cognition. This issue is not addressed in the current study.

could be implemented by creating an entity above the path level, which would represent an object moving along a trajectory. A moving-object model could favor the motion interpretations which were consistent with an object continuing a motion, without splitting, merging, or disappearing.

For now, we choose to resolve balanced activations by having the program avoid choosing a segment for which the endpoint is the same as an already active segment. Thus, once north is chosen, west will not be chosen, since north and west motions would terminate in the same endpoint.

There is, unfortunately, an order effect in the resolution of balanced activation. Since the program inputs features in a particular order, across the bitmap, and since the inputs are restricted to dots in a few specific locations, the program always processes the same dots first. This guides the order in which motion segments are added to the segment list, and this in turn leads the program to always choose the east/west direction for balanced input. We do not regard this limitation as a fundamental problem for our operational architecture. It can handle the resolution of balanced activation in many different ways that do not evidence a simple bias, which occurs with the current program.

3.3. Self-Organizing Systems

We have confined our discussion to this point to situations in which the input may reasonably be sorted into distinct features. This is not always the case. In particular, the mechanisms we have created may need to be modified in order to deal with overlapping percepts (Figure 21). Consider the following input:

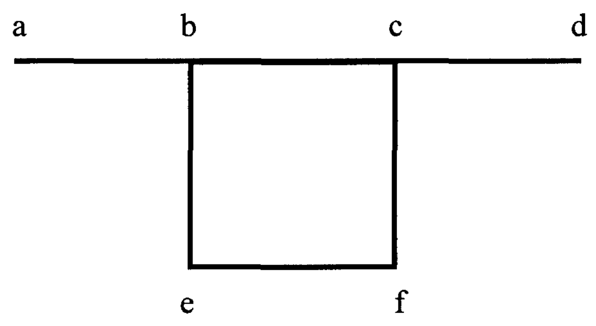


Figure 21. Overlapping Percepts.

Here, we have two potential overlapping percepts, square bcfe, and line segment ad. Our Cycle.java program would handle this input as follows: The initial input leads to the creation of four line objects: vertical line objects be and cf, and horizontal line objects ef and ad. Note that a potential line segment, bc, is not recognized as an object per se, but is subsumed under the larger line object, ad. After the square object is recognized, the square feeds activation to its component line segments, be, ef, cf, and bc. At this point, the program creates a distinct line object to represent bc. The square does not provide reinforcement to the entire segment, ad, but only to the segment bc which forms part of the square.

Several questions arise. When bc is created, should it inherit any existing level of activation from segment ad? (Our current program does provide for inheritance of activation.) Suppose subsequent processing depends on the presence of segment ad.

Should the level of activation of ad receive any boost from a higher level of activation for segment bc? That is, should the recognition of bc support the recognition of the larger line segment? Suppose we later inhibit either ad or bc; should this inhibit the overlapping segment, as well?

In order to handle a figure like the Necker cube, the Cycle.java program must be able to consolidate line segments into larger lines, as well as tolerating interruptions such as corners and crossing lines. At the same time, our definition of the components of a square defines the sides as excluding the corners, because some squares have corner elements which preclude a line interpretation. In particular, the Necker cube includes interior diagonal corners which appear as squares, instead of lines; see locations (2,16) and (16,2) in the bitmap b1. The combination of line consolidation and corner exclusion in square sides causes an overlap problem in the perception of a simple square. The initial input leads to a perception of the entire line segment, including the corners; but the square itself is defined with segments which exclude corners. Top-down activation of the component segments of the square thus activates only part of the initially perceived line segment. This result can be seen in the additional line segments listed under the master object list at the conclusion of the regularization output run.

Many psychologists assume that people perceive multiple percepts simultaneously by developing firing patterns in which several neurons, corresponding to a percept, begin to fire at the same rate. Different rates encode different percepts. The development of synchronized firing patterns from diverse input is called self-organization. In this section, we describe a simple, linear mechanism that can develop synchronization for one-dimensional input. We discuss how this mechanism might eventually be elaborated to handle two-dimensional input.

We create a third program, Domain.java, to implement the self-organization mechanism. Input is limited to single elements which occur at particular times; no spatial information is specified. Self-organization occurs as the elements are drawn into firing at various times. Input is represented by a data file, each line corresponding to a unit of time. We use letters to represent simple perceptual input. Each node reacts to one letter.

Nodes have an input code (the letter they reflect), an activation threshold, and an activation level. Repeated perceptual input causes activation to exceed threshold, and the node fires. When a node fires, it spreads activation to its neighbors, and resets its own activation to 0. The activation threshold is lowered, and the node is inhibited from firing for an interval, delta.

Spreading activation takes no time. (A spreading delay variable is provided in the code, however, delaying spreading activation may cause some timing problems.) A firing node may cause a neighbor node to also fire. (Currently, the model only spreads along chains of length 1.) When nodes co-fire, we strengthen the links between them. This encourages the same groups of nodes to co-fire again, even given only partial input.

A set of co-firing nodes is called a collocation. The model maintains an awareness of which collocations exist at each point in time. The time line is taken as a cycle, repeating after time=delta, where delta is the node refresh interval. For example, if delta is 5, then an elapsed time of 7 is considered the same as an elapsed time of 2. A node firing at time 7 is firing at the "same" time as a node firing at time 2.

When a collocation fires, its firing time may be reset by assimilation to other collocations which are nearby in time. A preceding collocation will encourage the current one to speed up its firing interval, while a following collocation will encourage the current one to slow down. This is the process by which nodes can self-organize into larger concepts.

In order to have influence, the neighboring collocation must occur within a temporal interval, r, which we consider to be the range of assimilation. (Together, delta and r determine the number of possible distinct collocations.) If there are both preceding and following collocations, we consider the distance and the number of nodes in the collocations in order to determine which has a stronger influence. (We call this "gravitation.") The amount of speeding up or slowing down, however, is not sensitive to the nature of the influence; we always speed up or slow down by a fixed assimilation amount.

When this program is run, closely spaced nodes gradually come together in time, showing that this simple process of nodes adjusting each other's firing times can lead to synchronization. Nodes b and c are input at time 1.0 and again at time 2.0, building up enough activation to fire at time 2.0. Activation spreads along a pre-coded b-a link, to partially activation node a. Nodes b and c cannot fire again until the domain interval, delta, has passed. Delta is set to 5, so b and c fire again at time 7.0, which is equivalent to 2.0. Node a cannot fire without additional activation. Continued b,c input would eventually cause node a to fire, joining the b,c collocation at time 2.0.

At time 4.0, new nodes l, m, n, o, and p are input. At time 5.0, which is equivalent to time 0.0, we again see nodes l, m, and n. These nodes fire. The existing b,c collocation at time 2.0 is close enough to affect the new collocation, and the firing time for collocation l,m,n is shifted from 0.0 to 0.2.

Next, at time 6.0, which is equivalent to time 1.0, we see nodes o and p. Previous activation and current activation allow these nodes to fire. This new o,p collocation at time 1.0 lies within range of both l,m,n at 0.0 and b,c at 2.0.[15] The larger collocation exerts more influence, and the firing time for collocation o,p is shifted from 1.0 to 0.8. Eventually, we would expect to see collocation o,p merge in firing time with collocation l,m,n, as these collocations mutually attract each other.

We suggest that this one-dimensional synchronization process could be extended to provide synchronization of two-dimensional input. Consider a partial organization mechanism, in which basic objects form a scaffolding for synchronizing input:

Certain simple objects (straight lines, e.g.) are learned with strong associations. These strongly-learned objects are easily recognized, even when input is asynchronous, and form the basis for organizing other input. If the input features for a basic object are asynchronously firing within a certain temporal range, smaller than the object decay rate, then the activation of these variously-timed inputs may accumulate sufficiently to cause recognition of the basic object. In effect, the system expects to see certain basic objects in any input.

Once some basic objects are recognized, those object nodes spread activation to their typical component features. This parallels the regularization of input discussed above; in this case, however, we do not activate a missing piece, but rather, nudge a

[15] The shift in firing from 0.0 to 0.2 for l,m,n would take effect on the next firing cycle, delaying l, m, and n from firing until time 0.2. The calculation for collocation o,p, however, is based on the most recent firing time for collocation l,m,n, which is still 0.0.

nearby piece into co-firing with the group. A spatial organization routine may then be employed to bring adjacent features into alignment with the developing groups.

4.0 GENERAL DISCUSSION AND CONCLUSION

The DCOG framework inherently addresses cognitive formation, activation, and use in an environmental context. Context is always changing and certain elements may at any point in time be regarded as noise, relative to the desired cognitive understanding of an actor. We can summarize the results of this investigation, looking at the coordination of activation decay with dual coding in a knowledge-level memory system, in terms of how the cognitive mind is able to deal with a noisy context. Our results show that by using separate activation decay time constants associated with different elements of the dual-coding structure, we were able to model perceptual cognition under three types of noisy context conditions. The first demonstration addressed concept build up to achieve the regularization of (noisy) incomplete input to yield knowledge-level percept recognition. The longer time constant for object elements in the dual-code structure provided activation persistence that enabled memory to fill in the missing data of a presented target, resulting in the perceptual cognition of the target as a whole object. The second demonstration addressed the noise induced by a target object that admits multiple interpretations; hence it is ambiguous. Two types of (noisy) ambiguous target objects were investigated. In the first case, the target was fashioned from form-based cues (i.e. a static object defined by spatially localized contrast.). In the second case, the target depended on motion for its existence (i.e. an apparent direction of movement of space-time element change). For both types of environmental input, the perceptual memory system was able to invoke human-like cognitive recognition, using the same type of activation decay time constants in coordination with the dual-coding memory structure. The third demonstration addressed the noise induced by overlapping target objects in an environment. Here the question is: how does the knowledge-level system discover the embedded object? Given the current work activity of an actor, the embedded object might be the target object of interest. Therefore the system must be able to identify the separate objects and aid the actor in orienting toward the one that is of use in this context. We addressed this problem by illustrating a simple self-organizing mechanism that can discover the embedded target. Taken as a whole, these results indicate that a simple storage structure and a small set of simple mechanisms are able to model important aspects of perceptual cognition. The demonstration showing how they could account for perceptual hysteresis adds additional support to this claim. Furthermore, the same structures and mechanisms define the core for recognition-based symbolic cognition, as outlined previously by Eggleston and McCreight (2006).

On logical ground, it seems reasonable that the persistence of activation would be longer for concept elements relative to feature elements. A feature element equates to an active feature node in the DCOG memory system. A concept element equates to a set of tightly connected, co-active feature nodes. Because activation strength tends to grow in time over the co-active set and because once a concept is activated it will tend to persist even as activation dies down (i.e. hysteresis effect), concepts once activated should take longer to drop below threshold. These factors of perceptual cognition suggest that a percept might fade out in fragments. In studies that have controlled for eye-movements (including micro nystagmus), an object image appears to fade out in fragments, as we would predict (e.g., Pritchard, 1961). The fading process is not at the retinal level (i.e., it is not a simple afterimage decay effect) but must occur at a higher brain level (Arend, 1973).

Dual coding is a key property of the DCOG memory system. In this study of dual-coding in the perceptual domain, we found it useful to introduce a mid-size element in the knowledge structure. This was true for both form-based perceptual recognition and motion-based perceptual recognition. The mid-size structure is expressed in memory as the co-activation of a tightly coupled set of feature nodes. This is the same pattern used to define a concept, and thus follows the dual-coding schema. It is only different in terms of the number of features included in the tightly coupled subnet. A concept has larger scope and will have more features in its subnet than the supporting mid-sized object subnet. Thus, even though three sizes of subnets have been labeled and made explicit in this study, they follow the same dual-coding principal developed in the original DCOG research.

It is not clear how many subnet sizes may be useful in a dual-coding memory system. Nor is it clear if there are differences in dual-coding across mind domains, and their associated differences in temporal processing. However, it may be possible to use constraints on dual-code structures as one way to differentiate, say, the perceptual domain from the symbolic domain. For example, the mid-size subnets, **object** (form-based percepts) and **segment** (motion-based percepts) used in this investigation are fully contained in the broader scope **concept**-size subnet. When co-active with the concept-size subnet, they encode (implicitly) a part-of relation. In our earlier treatment of the

DCOG memory model, we linked the concept tightly coupled subnet to an example-of relation. A concept is treated as a category. We also illustrated how more focused and smaller subnets could model the type-of relation. The DCOG memory structure for the type-of relation constitutes a subnet in the symbolic domain that is mid-sized like the object and segment subnets involved in percept recognition shown here. But there is also a difference. The type-of relation subnet included feature nodes that are not explicitly expressed in the feature set of the concept-size subnet. Other feature nodes in the backcloth are included in the type-of subnet. This suggests that a broader set of relations may be captured in the symbolic domain by the basic principal of a dual-coding memory structure. The perceptual domain may be constrained to only permit embedded subnet sizes within a percept.[16] Hence, this could be a reflection of the evolutionary drift that results in the emergence of a new mind domain that is similar to yet slightly different from other ones, which endow it with new capabilities.

The notion of attention is included in the DCOG memory system. However, we do not treat attention as a monolithic construct that identifies a single attentional system. Rather, we regard attention as a set of distributed, generally simple, mechanisms that are able to steer activation, both in a conscious and unconscious realm. For example, a simple threshold mechanism may serve in an attentional role in a given context. In an ambiguous figure context, as examined here, attention was expressed as a mid-size subnet of nodes that defined a 3-D object. The 3-D object dominates the interpretation of the Necker cube in terms of the perceived orientation that was manifested in the resulting percept recognition. This may reflect a property of the human perceptual domain. That

[16] Following this line of reasoning, this suggests that a concept-size memory subnet may be correlated with a low spatial frequency channel, with feature and other embedded subnet sizes corresponding to high spatial frequency channels.

is, it is biased toward a 3D organizational structure of scene content. Thus, the attentional mechanism might be something as simple as the 'automatic' activation of a well-learned feature subnet in the stereopsis region of the perceptual domain of the mind. The Necker cube creates difficulty because there are several competing (primitive) 3-D object size subnets that can be activated, each imposing a different viewpoint expressed in the concept-size subnet.

To date, we have not performed a systematic examination of attentional mechanisms in the DCOG framework, or even in its memory model. This is an area for further research.

This study adds additional strength to the notion of a dual-coding memory-based system as a foundation for knowledge-based cognition. The dual-coding structure has the unique property that it is bounded at the feature level of information (and hence bounded by the wetware), and yet it can support unbounded levels of abstraction based on relations that can be formed from other relations. Further, it provides a common way to model both the focal elements of cognition (concepts) and the context in which one or more concepts are simultaneously active. Hence, it naturally captures cognitive interpretation in context; the meaning of the situation and the focal concern are naturally connected. Another factor contributing to the flexibility and adaptive potential of the memory system derives from modeling a concept as the core of tightly coupled feature nodes. This rests at the heart of dual-coding and enables the simultaneous existence of a concept as a gestalt (whole entity) and its constituent parts. In our scheme, the whole is also different from the sum of its parts (i.e., the co-active tightly connected features yields the ‘whole’ object and the co-active loosely connected features are ‘part of’ the object and influence its (meaning) interpretation in the prevailing context).

It is clear that we did not perform a micro-level analysis of temporal properties in this study. Rather, the study serves to demonstrate in a general way that a simple temporal processing schema can be deployed with dual-coding to account for a range of perceptual capabilities. It suggests that a combination of simple spatially distributed and temporally distributed structures may be able to account for a highly flexible and adaptable memory system capable of human-scale cognition.

5.0. REFERENCES

Arend, L. (1973). Spatial differential and integral operations in human vision: Implications of stabilized retinal image fading. *Psychological Review* 80 374-395.

Atkin, R.H. (1974). *Mathematical Structure in Human Affairs*. London: Heinemann Educational Books.

Eggleston, R.G. (1986). *Apparent Motion and Prior Correspondence Effects in Visual Perception*. AAMRL Technical Report 86-027.

Eggleston, R.G. and McCreight, K.L. (2006). *Distributed Cognition: Foundations for an Associative Memory Model.* AFRL-TR-XXX-2006.

Eggleston, R.G., McCreight, K., and Young, M. (2005). Distributed Cognition and Situated Behavior. In Pew, R.W. and Gluck, K.A. (eds.) *Modeling Human Behavior with Integrated Cognitive Architectures: Comparison, Evaluation, and Validation.* NJ: Lawrence Erlbaum Associates

Eggleston, R.G., McCreight, K., and Young, M. (2001). Modeling Human Work through Distributed Cognition. *Proceedings of the Computer Generated Forces Conference*, Orlando, FL, Summer, 2001.

Eggleston, R.G., McCreight, K., and Young, M. (2000). Distributed Cognition: A New Type of Human Performance Model. *Proceedings of the AAAI Fall Symposium on Simulating Human Agents*, North Falmouth, MA, 3-5 November 2000.

Pritchard, R.M. (1961). Stablized Images on the Retina. *Scientific American*, 204, 72-78.

Rock, I. (1983). *The Logic of Perception.* Cambridge, MA: MIT Press.

Young, M. (1998). Computational Modeling of Memory: The Role of Long-Term Potentiation and Hebbian Synaptic Modification in Implicit and Schema Memory. *Dissertation Abstracts International*, Vol. 59/03-B.

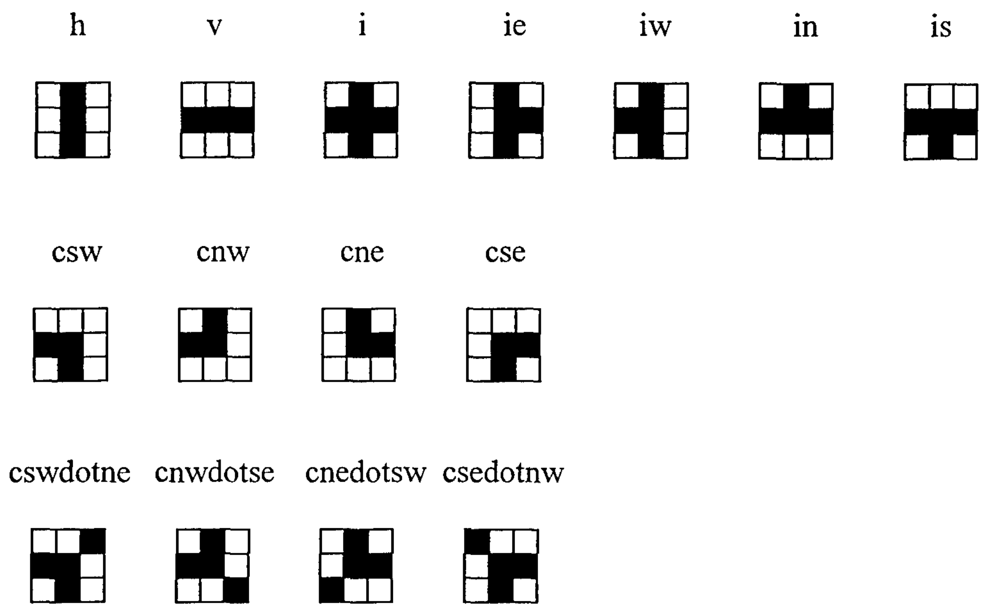
6.0 APPENDICES

The following section contains appendices that covers various aspects used in implementing the demonstrations discussed in the main body of the report.

Appendix A: Bit Grid Feature Detectors

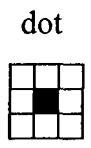
Here we describe the bit grids used to create the feature detector nodes in the programs Cycle.java and ApparentMotion.java.

Figure 1. Feature Detector Bit Grids



Note that the corners are defined in terms of their own orientation, not the orientation of a containing object. On a square, the SW corner of the square does not contain a csw bit grid; rather, since the corner opens to the NE, the bit grid in the SW corner of a square is a cne bit grid.

Figure 2. Bit Grid for Dot Feature Detector



The dot feature detector is used in ApparentMotion.java.

Table 1. Acronyms and Descriptions for Feature Detector Bit Grids

Acronym	Description
h	horizontal line
v	vertical line
i	intersection (cross)
ie	t-intersection with stem facing E
iw	t-intersection with stem facing W
in	t-intersection with stem facing N
is	t-intersection with stem facing S
csw	corner opening to the SW
cnw	corner opening to the NW
cne	corner opening to the NE
cse	corner opening to the SE
cswdotne	csw with added bit indicating diagonal
cnwdotse	cnw with added bit indicating diagonal
cnedotsw	cne with added bit indicating diagonal
csedotnw	cse with added bit indicating diagonal
dot	single bit (used in ApparentMotion.java)

Appendix B. Bit Grid Input for Cycle Program

Figure 1. Initial Bit Grid for Object Persistence (e1.bmp)

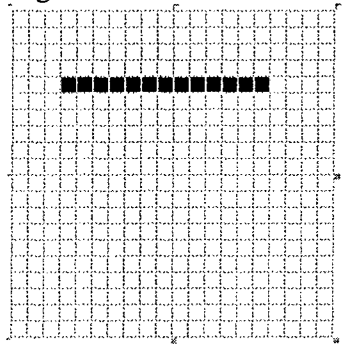


Figure 2. Subsequent Bit Grid for Object Persistence (e2.bmp, e3.bmp)

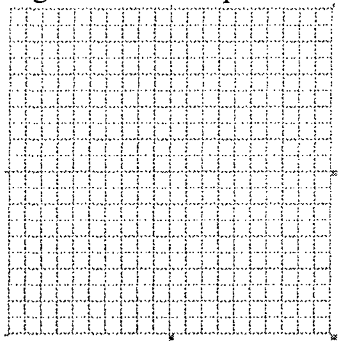


Figure 3. Bit Grid for Regularization of Input (b1.bmp)

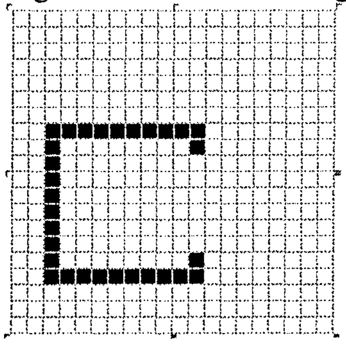
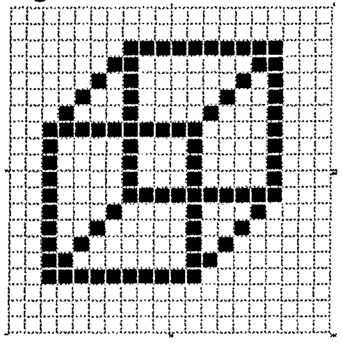
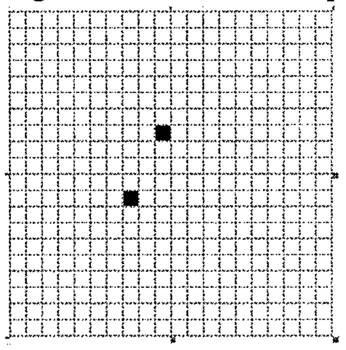


Figure 4. Bit Grid for Necker Cube (a1.bmp)



Appendix C. Bit Grid Input for ApparentMotion Program

Figure 1. Horizontal Displacement of 2


t1h2

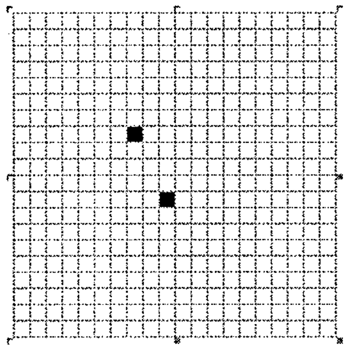
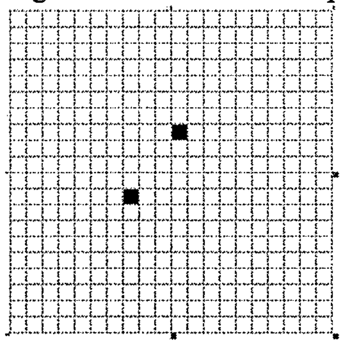

t2h2

Figure 2. Horizontal Displacement of 3


t1h3

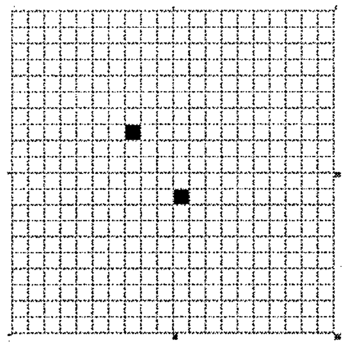
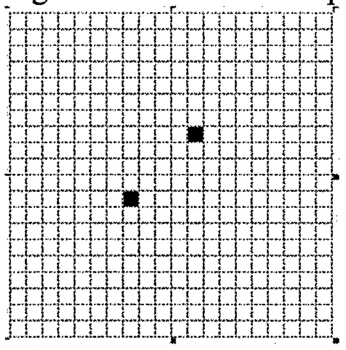

t2h3

Figure 3. Horizontal Displacement of 4


t1h4

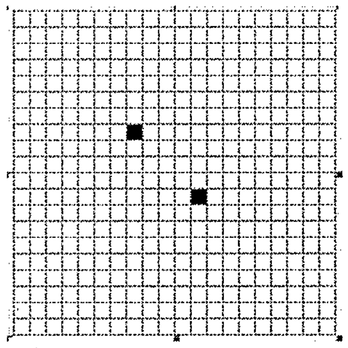
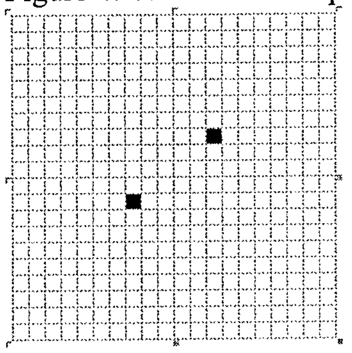
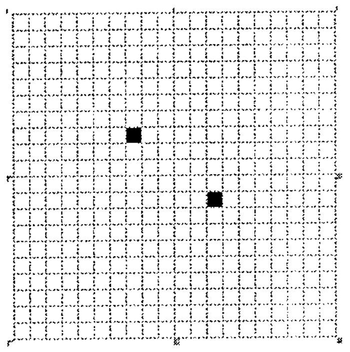

t2h4

Figure 4. Horizontal Displacement of 5

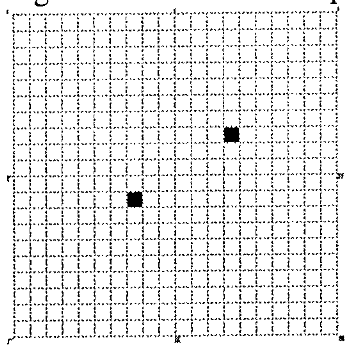


t1h5

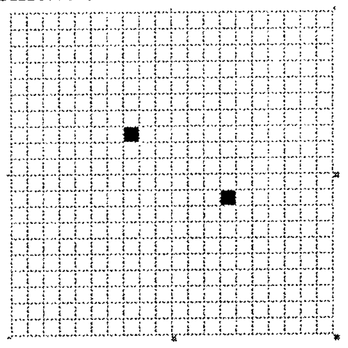


t2h5

Figure 5. Horizontal Displacement of 6

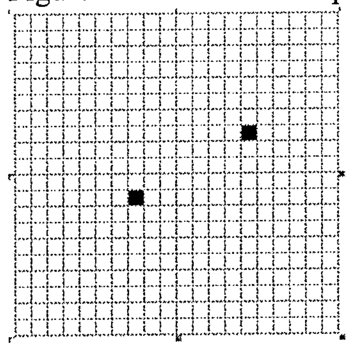


t1h6

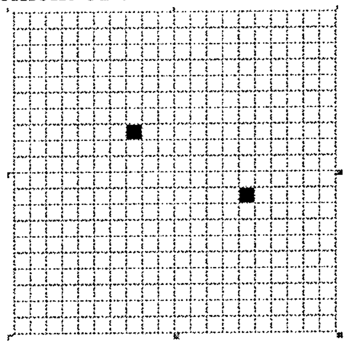


t2h6

Figure 6. Horizontal Displacement of 7

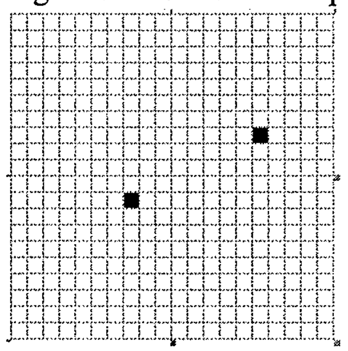


t1h7

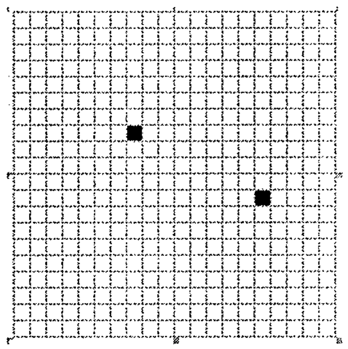


t2h7

Figure 7. Horizontal Displacement of 8

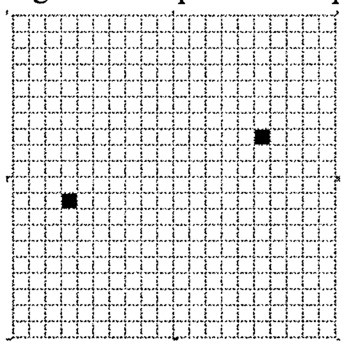


t1h8

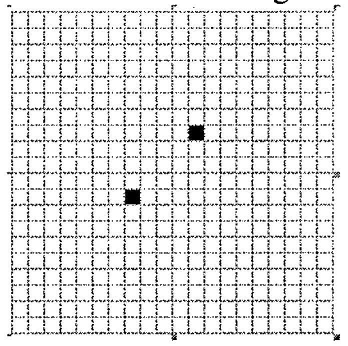


t2h8

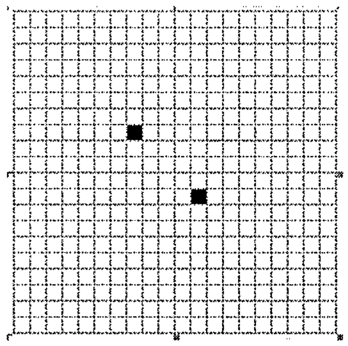
Figure 8. Sequence of Input for Horizontal Priming



t0h

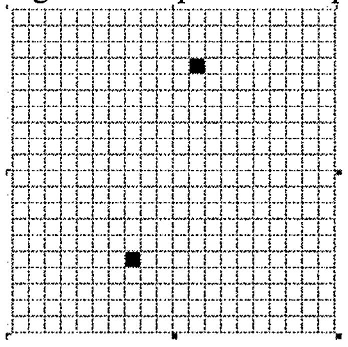


t1h4

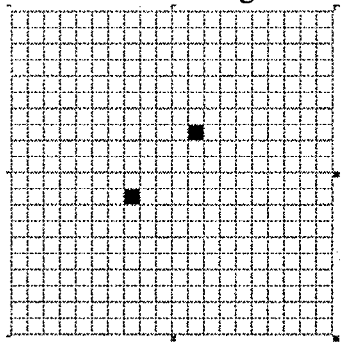


t2h4

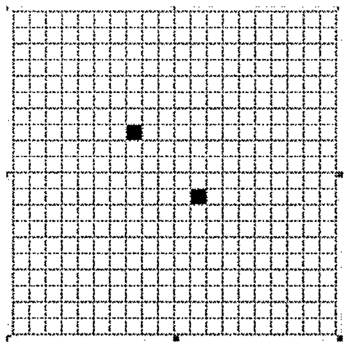
Figure 9. Sequence of Input for Vertical Priming



t0v



t1h4



t2h4

Appendix D. Data File for Domain.java

```
b,c
b,c
b,c
l,m,n,o,p
l,m,n
o,p
```

Appendix E. Cycle.java Output for Changing Input

```
=================
decaying feature activation at time 1.0
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 1.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 1.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 1.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 1.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 1.0
master concept list []
decaying feature activation at time 2.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 2.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 2.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 2.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 2.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 2.0
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$CornerSE@19f953d, Cycle$CornerNE@1fee6fc, Cycle$CornerSW@1eed786,
Cycle$CornerNW@187aeca]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$CornerSE@19f953d, Cycle$CornerNE@1fee6fc, Cycle$CornerSW@1eed786,
Cycle$CornerNW@187aeca]
lateral activation: percepti.type lineh
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
decaying object activation at time 2.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 3.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 1.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 1.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 1.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 1.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 1.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 1.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 1.0
```

```
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 1.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 1.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 1.0
master concept list []
decaying feature activation at time 4.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 2.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 2.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 2.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 2.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 2.0
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$CornerSE@19f953d, Cycle$CornerNE@1fee6fc, Cycle$CornerSW@1eed786,
Cycle$CornerNW@187aeca]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$CornerSE@19f953d, Cycle$CornerNE@1fee6fc, Cycle$CornerSW@1eed786,
Cycle$CornerNW@187aeca]
lateral activation: percepti.type lineh
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
decaying object activation at time 4.0
processing concepts at time 4.0
master concept list []
master concept list []
time=4.0, at resolve buffer conflicts with []
list of concepts over threshold becomes []
decaying concept activation at time 4.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 5.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
```

```
master concept list []
decaying feature activation at time 6.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
at check cube a
with active objects []
at check cube b
with active objects []
decaying object activation at time 6.0
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 7.0
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 8.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
at check cube a
with active objects []
at check cube b
with active objects []
decaying object activation at time 8.0
processing concepts at time 8.0
master concept list []
master concept list []
time=8.0, at resolve buffer conflicts with []
list of concepts over threshold becomes []
decaying concept activation at time 8.0
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 9.0
```

```
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 10.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
at check cube a
with active objects []
at check cube b
with active objects []
decaying object activation at time 10.0
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 11.0
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
decaying feature activation at time 12.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
at check cube a
with active objects []
at check cube b
with active objects []
decaying object activation at time 12.0
processing concepts at time 12.0
master concept list []
master concept list []
time=12.0, at resolve buffer conflicts with []
list of concepts over threshold becomes []
decaying concept activation at time 12.0
active objects are
master object list
```

```
Cycle$HorizontalLine@1cd2e5f lineh (2,4) act: 0.0
Cycle$CornerSE@19f953d cornerse (3,4) act: 0.0
Cycle$CornerNE@1fee6fc cornerne (3,4) act: 0.0
Cycle$CornerSW@1eed786 cornersw (15,4) act: 0.0
Cycle$CornerNW@187aeca cornernw (15,4) act: 0.0
master concept list []
```

Appendix F. Cycle.java Output for Regularization of Input

```
=================
starting regularization of input example
decaying feature activation at time 1.0
active objects are
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.0
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.0
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.0
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.0
master concept list []
decaying feature activation at time 2.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 2.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 2.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 2.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 2.0
Cycle$CornerNE@187aeca cornerne (2,16) act: 2.0
Cycle$CornerSW@e48e1b cornersw (11,7) act: 2.0
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 2.0
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1]
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
decaying object activation at time 2.0
decaying object activation to 0.775
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.0
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.0
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.0
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.0
```

```
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.0
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.0
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.0
Cycle$Square@18a992f square (2,16) act: 0.775
master concept list []
decaying feature activation at time 3.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.0
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.0
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.0
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.0
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.0
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.0
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.0
Cycle$Square@18a992f square (2,16) act: 0.775
master concept list []
decaying feature activation at time 4.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 2.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 2.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 2.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 2.0
Cycle$CornerNE@187aeca cornerne (2,16) act: 2.0
Cycle$CornerSW@e48e1b cornersw (11,7) act: 2.0
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 2.0
Cycle$Square@18a992f square (2,16) act: 0.775
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f]
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type square
decaying object activation at time 4.0
decaying object activation to 0.775
processing concepts at time 4.0
master concept list []
master concept list []
```

```
time=4.0, at resolve buffer conflicts with []
list of concepts over threshold becomes []
decaying concept activation at time 4.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.775
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.1
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 5.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.775
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.1
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 6.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 2.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 2.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 2.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 2.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 2.1
```

```
Cycle$CornerSW@e48e1b cornersw (11,7) act: 2.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 2.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.1
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f]
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type square
decaying object activation at time 6.0
decaying object activation to 0.775
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.775
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.2
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 7.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
```

```
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.775
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.2
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 8.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 2.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 2.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 2.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 2.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 2.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 2.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 2.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.2
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f]
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type square
decaying object activation at time 8.0
decaying object activation to 0.775
decaying object activation to 0.20000000000000004
processing concepts at time 8.0
master concept list []
master concept list []
time=8.0, at resolve buffer conflicts with []
list of concepts over threshold becomes []
```

```
decaying concept activation at time 8.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.775
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.20000000000000004
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 9.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.775
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 1.2
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 10.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 2.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 2.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 2.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 2.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 2.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 2.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 2.1
```

```
Cycle$Square@18a992f square (2,16) act: 0.775
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 1.2
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f,
Cycle$VerticalLine@dc8569]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f,
Cycle$VerticalLine@dc8569]
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type square
lateral activation: percepti.type linev
decaying object activation at time 10.0
decaying object activation to 0.9
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.9
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.1
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.9
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.1
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 11.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
```

```
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.9
Cycle$VerticalLine@dc8569 linev (11,8) act: 1.1
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 1.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 1.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 1.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 1.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 1.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 1.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 1.1
Cycle$Square@18a992f square (2,16) act: 0.9
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 1.1
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
decaying feature activation at time 12.0
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 2.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 2.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 2.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 2.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 2.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 2.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 2.1
Cycle$Square@18a992f square (2,16) act: 0.9
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 1.1
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
at check cube a
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f,
Cycle$VerticalLine@dc8569]
at check cube b
with active objects [Cycle$HorizontalLine@1cd2e5f,
Cycle$HorizontalLine@19f953d, Cycle$VerticalLine@1fee6fc,
Cycle$CornerSE@1eed786, Cycle$CornerNE@187aeca, Cycle$CornerSW@e48e1b,
Cycle$CornerNW@12dacd1, Cycle$Square@18a992f,
Cycle$VerticalLine@dc8569]
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type square
lateral activation: percepti.type linev
decaying object activation at time 12.0
decaying object activation to 0.9
```

```
processing concepts at time 12.0
master concept list []
master concept list []
time=12.0, at resolve buffer conflicts with []
list of concepts over threshold becomes []
decaying concept activation at time 12.0
active objects are
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.9
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.1
master object list
Cycle$HorizontalLine@1cd2e5f lineh (2,7) act: 0.0
Cycle$HorizontalLine@19f953d lineh (2,16) act: 0.0
Cycle$VerticalLine@1fee6fc linev (2,7) act: 0.0
Cycle$CornerSE@1eed786 cornerse (2,7) act: 0.1
Cycle$CornerNE@187aeca cornerne (2,16) act: 0.1
Cycle$CornerSW@e48e1b cornersw (11,7) act: 0.1
Cycle$CornerNW@12dacd1 cornernw (11,16) act: 0.1
Cycle$Square@18a992f square (2,16) act: 0.9
Cycle$HorizontalLine@4f1d0d lineh (3,7) act: 0.1
Cycle$HorizontalLine@1fc4bec lineh (3,16) act: 0.1
Cycle$VerticalLine@dc8569 linev (11,8) act: 0.1
Cycle$VerticalLine@1bab50a linev (2,8) act: 0.1
master concept list []
```

Appendix G. Cycle.java Output for Necker Cube with Attention

```
~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~
Necker Cube with attention trial
conceptTime is 0.0
old time is 0.0
active object list is []
---------------------
with general attention to objects of type corner ne 3d
))) time = 0.0:  active concepts []
master concept list []
active concepts are []
decaying feature activation at time 1.0
active objects are
))) time = 1.0:  active concepts []
master concept list []
active concepts are []
decaying feature activation at time 2.0
master object list
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 2.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 2.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 2.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 2.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 2.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 2.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 2.0
Cycle$VerticalLine@42719c linev (16,3) act: 2.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 2.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 2.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 2.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 2.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 2.0
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 2.0
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 2.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 2.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 2.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 2.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 2.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 2.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 2.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 2.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 2.0
Cycle$CornerNE@7c6768 cornerne (7,11) act: 2.0
Cycle$CornerSW@1690726 cornersw (11,7) act: 2.0
Cycle$CornerNW@5483cd cornernw (11,16) act: 2.0
Cycle$CornerSE@9931f5 cornerse (12,5) act: 2.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 2.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 2.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 2.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 2.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 2.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 2.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 2.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 2.0
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 2.0
```

```
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 2.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 2.0
at check cube a
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5]
at check cube b
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5]
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw3d
lateral activation: percepti.type cornerne3d
decaying object activation at time 2.0
decaying object activation to 0.9
decaying object activation to 0.9
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 0.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 0.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 0.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 0.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 0.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 0.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 0.0
Cycle$VerticalLine@42719c linev (16,3) act: 0.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 0.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 0.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 0.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 0.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 0.0
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 0.0
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 0.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 0.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 0.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 0.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 0.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 0.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 0.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 0.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 0.0
Cycle$CornerNE@7c6768 cornerne (7,11) act: 0.0
Cycle$CornerSW@1690726 cornersw (11,7) act: 0.0
Cycle$CornerNW@5483cd cornernw (11,16) act: 0.0
Cycle$CornerSE@9931f5 cornerse (12,5) act: 0.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 0.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 0.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 0.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 0.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 0.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 0.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 0.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 0.0
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 0.0
```

```
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 0.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 0.0
))) time = 2.0:  active concepts []
master concept list []
active concepts are []
decaying feature activation at time 3.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 1.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 1.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 1.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 1.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 1.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 1.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 1.0
Cycle$VerticalLine@42719c linev (16,3) act: 1.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 1.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 1.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 1.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 1.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 1.0
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 1.0
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 1.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 1.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 1.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 1.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 1.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 1.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 1.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 1.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 1.0
Cycle$CornerNE@7c6768 cornerne (7,11) act: 1.0
Cycle$CornerSW@1690726 cornersw (11,7) act: 1.0
Cycle$CornerNW@5483cd cornernw (11,16) act: 1.0
Cycle$CornerSE@9931f5 cornerse (12,5) act: 1.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 1.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 1.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 1.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 1.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 1.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 1.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 1.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 1.0
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 1.0
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 1.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 1.0
))) time = 3.0:  active concepts []
master concept list []
active concepts are []
decaying feature activation at time 4.0
master object list
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 2.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 2.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 2.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 2.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 2.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 2.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 2.0
```

```
Cycle$VerticalLine@42719c linev (16,3) act: 2.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 2.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 2.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 2.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 2.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 2.0
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 2.0
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 2.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 2.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 2.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 2.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 2.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 2.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 2.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 2.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 2.0
Cycle$CornerNE@7c6768 cornerne (7,11) act: 2.0
Cycle$CornerSW@1690726 cornersw (11,7) act: 2.0
Cycle$CornerNW@5483cd cornernw (11,16) act: 2.0
Cycle$CornerSE@9931f5 cornerse (12,5) act: 2.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 2.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 2.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 2.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 2.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 2.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 2.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 2.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 2.0
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 2.0
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 2.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 2.0
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 0.9
at check cube a
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
added possible cube Cycle$CubeA@1cfb549 with act 0.0 and off=false
cubek is Cycle$CubeA@1cfb549711
cubek parts list [Cycle$Square@b162d5, Cycle$CornerNE3D@1034bb5,
Cycle$HorizontalLine@186d4c1, Cycle$VerticalLine@f9f9d8,
Cycle$DiagonalLineSWNE@1820dda, Cycle$DiagonalLineSWNE@15b7986]
found part square at 7,11
found part cornerne3d at 7,11
found part lineh at 3,16
found part linev at 2,8
found part diagswne at 3,6
found part diagswne at 12,15
proportion matched is 1.0
matched 1.0, recognizing CubeA
at check cube b
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
cubek is Cycle$CubeB@87816d216
cubek parts list [Cycle$Square@15f5897, Cycle$CornerSW3D@a0dcd9,
Cycle$HorizontalLine@422ede, Cycle$VerticalLine@112f614,
Cycle$DiagonalLineSWNE@1d9dc39, Cycle$DiagonalLineSWNE@93dcd]
found part square at 2,16
found part cornersw3d at 11,7
found part lineh at 3,7
found part linev at 11,8
found part diagswne at 12,15
found part diagswne at 3,6
matched 6.0 parts
matched 1.0, recognizing CubeB
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw3d
lateral activation: percepti.type cornerne3d
lateral activation: percepti.type square
lateral activation: percepti.type square
decaying object activation at time 4.0
decaying object activation to 0.9
decaying object activation to 0.9
processing concepts at time 4.0
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.5 off =false
Cycle$CubeB@87816d cubeB (2,16) act: 0.5 off =false
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
potential object of type cubeA
with activation 0.5
concepti.off false
concept over threshold cubeA at 7,11
potential object of type cubeB
with activation 0.5
concepti.off false
concept over threshold cubeB at 2,16
time=4.0, at resolve buffer conflicts with [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
conflict between cubeA act 0.5 and cubeB act 0.5
activations equal
list of concepts over threshold becomes [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
decaying concept activation at time 4.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 0.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 0.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 0.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 0.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 0.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 0.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 0.0
Cycle$VerticalLine@42719c linev (16,3) act: 0.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 0.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 0.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 0.2
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 0.2
Cycle$CornerSE@18a992f cornerse (2,7) act: 0.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 0.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 0.0
```

```
Cycle$CornerSE@dc8569 cornerse (4,4) act: 0.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 0.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 0.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 0.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 0.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 0.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 0.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 0.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 0.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 0.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 0.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 0.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 0.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 0.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 0.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 0.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 0.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 0.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 0.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 0.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 0.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 0.2
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 0.2
Cycle$Square@15f5897 square (2,16) act: 1.1
Cycle$Square@b162d5 square (7,11) act: 1.1
))) time = 4.0:  active concepts [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 0.0 off =true
active concepts are [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0
Cycle$CubeB@87816d cubeB (2,16) act: 0.0
-------------
-------------
decaying feature activation at time 5.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 1.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 1.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 1.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 1.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 1.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 1.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 1.0
Cycle$VerticalLine@42719c linev (16,3) act: 1.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 1.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 1.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 1.2
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 1.2
Cycle$CornerSE@18a992f cornerse (2,7) act: 1.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 1.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 1.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 1.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 1.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 1.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 1.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 1.0
```

```
Cycle$CornerNW@f62373 cornernw (5,14) act: 1.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 1.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 1.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 1.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 1.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 1.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 1.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 1.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 1.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 1.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 1.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 1.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 1.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 1.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 1.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 1.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 1.2
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 1.7
Cycle$Square@15f5897 square (2,16) act: 1.1
Cycle$Square@b162d5 square (7,11) act: 1.1
))) time = 5.0:  active concepts [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.5 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 0.0 off =true
active concepts are [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.5
Cycle$CubeB@87816d cubeB (2,16) act: 0.0
decaying feature activation at time 6.0
master object list
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 2.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 2.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 2.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 2.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 2.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 2.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 2.0
Cycle$VerticalLine@42719c linev (16,3) act: 2.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 2.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 2.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 2.2
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 2.2
Cycle$CornerSE@18a992f cornerse (2,7) act: 2.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 2.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 2.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 2.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 2.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 2.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 2.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 2.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 2.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 2.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 2.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 2.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 2.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 2.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 2.0
```

```
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 2.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 2.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 2.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 2.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 2.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 2.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 2.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 2.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 2.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 2.2
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 2.7
Cycle$Square@15f5897 square (2,16) act: 1.1
Cycle$Square@b162d5 square (7,11) act: 1.1
Cycle$HorizontalLine@723d7c lineh (3,7) act: 0.30000000000000004
Cycle$HorizontalLine@22c95b lineh (3,16) act: 0.30000000000000004
Cycle$VerticalLine@1d1acd3 linev (11,8) act: 0.30000000000000004
Cycle$VerticalLine@a981ca linev (2,8) act: 0.30000000000000004
Cycle$HorizontalLine@8814e9 lineh (8,2) act: 0.1
Cycle$HorizontalLine@1503a3 lineh (8,11) act: 0.1
Cycle$VerticalLine@1a1c887 linev (16,3) act: 0.1
Cycle$VerticalLine@743399 linev (7,3) act: 0.1
Cycle$DiagonalLineSWNE@1d9dc39 diagswne (12,15) act: 0.2
Cycle$DiagonalLineSWNE@93dcd diagswne (3,6) act: 0.2
at check cube a
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
added possible cube Cycle$CubeA@e7b241 with act 0.0 and off=false
cubek is Cycle$CubeA@e7b241711
cubek parts list [Cycle$Square@b162d5, Cycle$CornerNE3D@1034bb5,
Cycle$HorizontalLine@167d940, Cycle$VerticalLine@e83912,
Cycle$DiagonalLineSWNE@1fae3c6, Cycle$DiagonalLineSWNE@7ffe01]
found part square at 7,11
found part cornerne3d at 7,11
found part lineh at 3,16
found part linev at 2,8
found part diagswne at 3,6
found part diagswne at 12,15
proportion matched is 1.0
matched 1.0, recognizing CubeA
at check cube b
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
cubek is Cycle$CubeB@fd13b5216
cubek parts list [Cycle$Square@15f5897, Cycle$CornerSW3D@a0dcd9,
Cycle$HorizontalLine@118f375, Cycle$VerticalLine@117a8bd,
Cycle$DiagonalLineSWNE@471e30, Cycle$DiagonalLineSWNE@10ef90c]
found part square at 2,16
found part cornersw3d at 11,7
found part lineh at 3,7
found part linev at 11,8
found part diagswne at 12,15
found part diagswne at 3,6
matched 6.0 parts
matched 1.0, recognizing CubeB
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw3d
lateral activation: percepti.type cornerne3d
lateral activation: percepti.type square
lateral activation: percepti.type square
decaying object activation at time 6.0
decaying object activation to 0.9
decaying object activation to 0.9
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 0.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 0.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 0.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 0.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 0.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 0.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 0.0
Cycle$VerticalLine@42719c linev (16,3) act: 0.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 0.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 0.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 0.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 0.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 0.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 0.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 0.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 0.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 0.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 0.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 0.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 0.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 0.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 0.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 0.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 0.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 0.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 0.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 0.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 0.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 0.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 0.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 0.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 0.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 0.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 0.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 0.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 0.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 0.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 0.0
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 0.9
))) time = 6.0:  active concepts [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 1.0 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 0.5 off =true
```

```
active concepts are [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 1.0
Cycle$CubeB@87816d cubeB (2,16) act: 0.5
decaying feature activation at time 7.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 1.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 1.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 1.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 1.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 1.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 1.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 1.0
Cycle$VerticalLine@42719c linev (16,3) act: 1.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 1.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 1.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 1.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 1.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 1.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 1.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 1.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 1.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 1.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 1.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 1.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 1.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 1.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 1.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 1.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 1.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 1.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 1.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 1.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 1.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 1.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 1.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 1.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 1.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 1.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 1.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 1.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 1.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 1.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 1.0
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 0.9
))) time = 7.0:  active concepts [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 1.0 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 0.5 off =true
active concepts are [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 1.0
Cycle$CubeB@87816d cubeB (2,16) act: 0.5
decaying feature activation at time 8.0
master object list
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 2.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 2.0
```

```
Cycle$HorizontalLine@187aeca lineh (3,16) act: 2.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 2.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 2.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 2.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 2.0
Cycle$VerticalLine@42719c linev (16,3) act: 2.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 2.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 2.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 2.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 2.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 2.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 2.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 2.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 2.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 2.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 2.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 2.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 2.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 2.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 2.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 2.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 2.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 2.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 2.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 2.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 2.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 2.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 2.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 2.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 2.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 2.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 2.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 2.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 2.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 2.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 2.0
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 0.9
Cycle$HorizontalLine@723d7c lineh (3,7) act: 0.1
Cycle$HorizontalLine@22c95b lineh (3,16) act: 0.1
Cycle$VerticalLine@1d1acd3 linev (11,8) act: 0.1
Cycle$VerticalLine@a981ca linev (2,8) act: 0.1
Cycle$HorizontalLine@8814e9 lineh (8,2) act: 0.1
Cycle$HorizontalLine@1503a3 lineh (8,11) act: 0.1
Cycle$VerticalLine@1a1c887 linev (16,3) act: 0.1
Cycle$VerticalLine@743399 linev (7,3) act: 0.1
Cycle$DiagonalLineSWNE@1d9dc39 diagswne (12,15) act: 0.2
Cycle$DiagonalLineSWNE@93dcd diagswne (3,6) act: 0.2
at check cube a
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
added possible cube Cycle$CubeA@a32b with act 0.0 and off=false
cubek is Cycle$CubeA@a32b711
cubek parts list [Cycle$Square@b162d5, Cycle$CornerNE3D@1034bb5,
Cycle$HorizontalLine@1d8957f, Cycle$VerticalLine@3ee284,
Cycle$DiagonalLineSWNE@8965fb, Cycle$DiagonalLineSWNE@867e89]
found part square at 7,11
found part cornerne3d at 7,11
found part lineh at 3,16
found part linev at 2,8
found part diagswne at 3,6
found part diagswne at 12,15
proportion matched is 1.0
matched 1.0, recognizing CubeA
at check cube b
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
cubek is Cycle$CubeB@1dd7056216
cubek parts list [Cycle$Square@15f5897, Cycle$CornerSW3D@a0dcd9,
Cycle$HorizontalLine@fa3ac1, Cycle$VerticalLine@276af2,
Cycle$DiagonalLineSWNE@1de3f2d, Cycle$DiagonalLineSWNE@5d173]
found part square at 2,16
found part cornersw3d at 11,7
found part lineh at 3,7
found part linev at 11,8
found part diagswne at 12,15
found part diagswne at 3,6
matched 6.0 parts
matched 1.0, recognizing CubeB
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw3d
lateral activation: percepti.type cornerne3d
lateral activation: percepti.type square
lateral activation: percepti.type square
decaying object activation at time 8.0
decaying object activation to 0.9
decaying object activation to 0.9
processing concepts at time 8.0
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 1.5 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 1.0 off =true
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
potential object of type cubeA
with activation 1.5
concepti.off false
concept over threshold cubeA at 7,11
potential object of type cubeB
with activation 1.0
concepti.off false
concept over threshold cubeB at 2,16
time=8.0, at resolve buffer conflicts with [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
conflict between cubeA act 1.5 and cubeB act 1.0
removing cubeB
list of concepts over threshold becomes [Cycle$CubeA@1cfb549]
decaying concept activation at time 8.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 0.0
```

```
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 0.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 0.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 0.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 0.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 0.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 0.0
Cycle$VerticalLine@42719c linev (16,3) act: 0.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 0.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 0.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 0.2
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 0.2
Cycle$CornerSE@18a992f cornerse (2,7) act: 0.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 0.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 0.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 0.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 0.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 0.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 0.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 0.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 0.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 0.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 0.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 0.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 0.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 0.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 0.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 0.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 0.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 0.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 0.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 0.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 0.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 0.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 0.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 0.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 0.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 0.2
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 1.1
))) time = 8.0:  active concepts [Cycle$CubeA@1cfb549]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 0.0 off =false
active concepts are [Cycle$CubeA@1cfb549]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0
-------------
-------------
decaying feature activation at time 9.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 1.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 1.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 1.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 1.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 1.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 1.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 1.0
Cycle$VerticalLine@42719c linev (16,3) act: 1.0
```

```
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 1.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 1.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 1.2
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 1.2
Cycle$CornerSE@18a992f cornerse (2,7) act: 1.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 1.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 1.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 1.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 1.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 1.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 1.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 1.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 1.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 1.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 1.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 1.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 1.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 1.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 1.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 1.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 1.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 1.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 1.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 1.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 1.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 1.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 1.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 1.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 1.5
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 1.2
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 1.1
))) time = 9.0:  active concepts [Cycle$CubeA@1cfb549]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 0.5 off =false
active concepts are [Cycle$CubeA@1cfb549]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0
decaying feature activation at time 10.0
master object list
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 2.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 2.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 2.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 2.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 2.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 2.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 2.0
Cycle$VerticalLine@42719c linev (16,3) act: 2.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 2.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 2.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 2.2
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 2.2
Cycle$CornerSE@18a992f cornerse (2,7) act: 2.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 2.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 2.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 2.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 2.0
```

```
Cycle$CornerSE@c3c749 cornerse (5,3) act: 2.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 2.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 2.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 2.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 2.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 2.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 2.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 2.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 2.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 2.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 2.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 2.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 2.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 2.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 2.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 2.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 2.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 2.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 2.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 2.5
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 2.2
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 1.1
Cycle$HorizontalLine@723d7c lineh (3,7) act: 0.1
Cycle$HorizontalLine@22c95b lineh (3,16) act: 0.30000000000000004
Cycle$VerticalLine@1d1acd3 linev (11,8) act: 0.1
Cycle$VerticalLine@a981ca linev (2,8) act: 0.30000000000000004
Cycle$HorizontalLine@8814e9 lineh (8,2) act: 0.1
Cycle$HorizontalLine@1503a3 lineh (8,11) act: 0.1
Cycle$VerticalLine@1a1c887 linev (16,3) act: 0.1
Cycle$VerticalLine@743399 linev (7,3) act: 0.1
Cycle$DiagonalLineSWNE@1d9dc39 diagswne (12,15) act: 0.2
Cycle$DiagonalLineSWNE@93dcd diagswne (3,6) act: 0.2
at check cube a
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
added possible cube Cycle$CubeA@cd2c3c with act 0.0 and off=false
cubek is Cycle$CubeA@cd2c3c711
cubek parts list [Cycle$Square@b162d5, Cycle$CornerNE3D@1034bb5,
Cycle$HorizontalLine@13582d, Cycle$VerticalLine@21b6d,
Cycle$DiagonalLineSWNE@56a499, Cycle$DiagonalLineSWNE@506411]
found part square at 7,11
found part cornerne3d at 7,11
found part lineh at 3,16
found part linev at 2,8
found part diagswne at 3,6
found part diagswne at 12,15
proportion matched is 1.0
matched 1.0, recognizing CubeA
at check cube b
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
cubek is Cycle$CubeB@1d99a4d216
cubek parts list [Cycle$Square@15f5897, Cycle$CornerSW3D@a0dcd9,
Cycle$HorizontalLine@12152e6, Cycle$VerticalLine@c9ba38,
Cycle$DiagonalLineSWNE@1e0be38, Cycle$DiagonalLineSWNE@1e859c0]
found part square at 2,16
found part cornersw3d at 11,7
found part lineh at 3,7
found part linev at 11,8
found part diagswne at 12,15
found part diagswne at 3,6
matched 6.0 parts
matched 1.0, recognizing CubeB
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw3d
lateral activation: percepti.type cornerne3d
lateral activation: percepti.type square
lateral activation: percepti.type square
decaying object activation at time 10.0
decaying object activation to 0.9
decaying object activation to 0.9
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 0.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 0.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 0.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 0.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 0.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 0.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 0.0
Cycle$VerticalLine@42719c linev (16,3) act: 0.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 0.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 0.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 0.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 0.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 0.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 0.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 0.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 0.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 0.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 0.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 0.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 0.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 0.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 0.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 0.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 0.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 0.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 0.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 0.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 0.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 0.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 0.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 0.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 0.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 0.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 0.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 0.1
```

```
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 0.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 0.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 0.0
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 0.9
))) time = 10.0:  active concepts [Cycle$CubeA@1cfb549]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.5 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 1.0 off =false
active concepts are [Cycle$CubeA@1cfb549]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.5
decaying feature activation at time 11.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 1.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 1.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 1.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 1.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 1.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 1.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 1.0
Cycle$VerticalLine@42719c linev (16,3) act: 1.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 1.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 1.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 1.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 1.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 1.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 1.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 1.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 1.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 1.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 1.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 1.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 1.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 1.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 1.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 1.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 1.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 1.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 1.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 1.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 1.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 1.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 1.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 1.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 1.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 1.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 1.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 1.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 1.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 1.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 1.0
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 0.9
))) time = 11.0:  active concepts [Cycle$CubeA@1cfb549]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.5 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 1.0 off =false
```

```
active concepts are [Cycle$CubeA@1cfb549]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.5
decaying feature activation at time 12.0
master object list
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 2.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 2.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 2.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 2.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 2.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 2.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 2.0
Cycle$VerticalLine@42719c linev (16,3) act: 2.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 2.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 2.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 2.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 2.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 2.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 2.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 2.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 2.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 2.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 2.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 2.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 2.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 2.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 2.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 2.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 2.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 2.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 2.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 2.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 2.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 2.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 2.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 2.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 2.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 2.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 2.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 2.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 2.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 2.0
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 2.0
Cycle$Square@15f5897 square (2,16) act: 0.9
Cycle$Square@b162d5 square (7,11) act: 0.9
Cycle$HorizontalLine@723d7c lineh (3,7) act: 0.1
Cycle$HorizontalLine@22c95b lineh (3,16) act: 0.1
Cycle$VerticalLine@1d1acd3 linev (11,8) act: 0.1
Cycle$VerticalLine@a981ca linev (2,8) act: 0.1
Cycle$HorizontalLine@8814e9 lineh (8,2) act: 0.1
Cycle$HorizontalLine@1503a3 lineh (8,11) act: 0.1
Cycle$VerticalLine@1a1c887 linev (16,3) act: 0.1
Cycle$VerticalLine@743399 linev (7,3) act: 0.1
Cycle$DiagonalLineSWNE@1d9dc39 diagswne (12,15) act: 0.2
Cycle$DiagonalLineSWNE@93dcd diagswne (3,6) act: 0.2
at check cube a
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
added possible cube Cycle$CubeA@16a9d42 with act 0.0 and off=false
cubek is Cycle$CubeA@16a9d42711
cubek parts list [Cycle$Square@b162d5, Cycle$CornerNE3D@1034bb5,
Cycle$HorizontalLine@7a84e4, Cycle$VerticalLine@1aaa14a,
Cycle$DiagonalLineSWNE@1430b5c, Cycle$DiagonalLineSWNE@9ed927]
found part square at 7,11
found part cornerne3d at 7,11
found part lineh at 3,16
found part linev at 2,8
found part diagswne at 3,6
found part diagswne at 12,15
proportion matched is 1.0
matched 1.0, recognizing CubeA
at check cube b
with active objects [Cycle$HorizontalLine@1fee6fc,
Cycle$HorizontalLine@1eed786, Cycle$HorizontalLine@187aeca,
Cycle$HorizontalLine@e48e1b, Cycle$VerticalLine@12dacd1,
Cycle$VerticalLine@1ad086a, Cycle$VerticalLine@10385c1,
Cycle$VerticalLine@42719c, Cycle$DiagonalLineSWNE@30c221,
Cycle$DiagonalLineSWNE@119298d, Cycle$DiagonalLineSWNE@f72617,
Cycle$DiagonalLineSWNE@1e5e2c3, Cycle$CornerSE@18a992f,
Cycle$CornerNE@4f1d0d, Cycle$CornerSE@1fc4bec, Cycle$CornerSE@dc8569,
Cycle$CornerSE@1bab50a, Cycle$CornerSE@c3c749, Cycle$CornerNW@6e1408,
Cycle$CornerSE@e53108, Cycle$CornerNW@f62373, Cycle$CornerNW@19189e1,
Cycle$CornerSE@1f33675, Cycle$CornerNE@7c6768, Cycle$CornerSW@1690726,
Cycle$CornerNW@5483cd, Cycle$CornerSE@9931f5, Cycle$CornerSE@19ee1ac,
Cycle$CornerNW@1f1fba0, Cycle$CornerSE@1befab0, Cycle$CornerNW@13c5982,
Cycle$CornerNW@1186fab, Cycle$CornerNW@14b7453, Cycle$CornerNW@c21495,
Cycle$CornerSW@1d5550d, Cycle$CornerNW@c2ea3f, Cycle$CornerSW3D@a0dcd9,
Cycle$CornerNE3D@1034bb5, Cycle$Square@15f5897, Cycle$Square@b162d5]
cubek is Cycle$CubeB@c2a132216
cubek parts list [Cycle$Square@15f5897, Cycle$CornerSW3D@a0dcd9,
Cycle$HorizontalLine@1e51060, Cycle$VerticalLine@19616c7,
Cycle$DiagonalLineSWNE@b166b5, Cycle$DiagonalLineSWNE@cdfc9c]
found part square at 2,16
found part cornersw3d at 11,7
found part lineh at 3,7
found part linev at 11,8
found part diagswne at 12,15
found part diagswne at 3,6
matched 6.0 parts
matched 1.0, recognizing CubeB
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type lineh
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type linev
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type diagswne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerne
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornerse
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw
lateral activation: percepti.type cornernw
lateral activation: percepti.type cornersw3d
lateral activation: percepti.type cornerne3d
lateral activation: percepti.type square
lateral activation: percepti.type square
decaying object activation at time 12.0
decaying object activation to 0.9
decaying object activation to 0.9
processing concepts at time 12.0
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 1.0 off =true
Cycle$CubeB@87816d cubeB (2,16) act: 1.5 off =false
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
potential object of type cubeA
with activation 1.0
concepti.off false
concept over threshold cubeA at 7,11
potential object of type cubeB
with activation 1.5
concepti.off false
concept over threshold cubeB at 2,16
time=12.0, at resolve buffer conflicts with [Cycle$CubeA@1cfb549,
Cycle$CubeB@87816d]
```

```
conflict between cubeA act 1.0 and cubeB act 1.5
removing cubeA
list of concepts over threshold becomes [Cycle$CubeB@87816d]
decaying concept activation at time 12.0
active objects are
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 0.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 0.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 0.0
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 0.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 0.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 0.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 0.0
Cycle$VerticalLine@42719c linev (16,3) act: 0.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 0.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 0.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 0.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 0.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 0.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 0.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 0.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 0.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 0.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 0.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 0.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 0.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 0.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 0.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 0.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 0.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 0.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 0.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 0.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 0.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 0.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 0.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 0.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 0.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 0.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 0.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 0.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 0.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 0.2
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 0.0
Cycle$Square@15f5897 square (2,16) act: 1.1
Cycle$Square@b162d5 square (7,11) act: 0.9
))) time = 12.0:  active concepts [Cycle$CubeB@87816d]
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0 off =false
Cycle$CubeB@87816d cubeB (2,16) act: 0.0 off =true
active concepts are [Cycle$CubeB@87816d]
Cycle$CubeB@87816d cubeB (2,16) act: 0.0
---------------
with general attention to objects of type corner ne 3d
master object list
Cycle$HorizontalLine@1fee6fc lineh (3,7) act: 0.0
Cycle$HorizontalLine@1eed786 lineh (7,11) act: 0.0
Cycle$HorizontalLine@187aeca lineh (3,16) act: 0.0
```

```
Cycle$HorizontalLine@e48e1b lineh (7,2) act: 0.0
Cycle$VerticalLine@12dacd1 linev (7,3) act: 0.0
Cycle$VerticalLine@1ad086a linev (11,7) act: 0.0
Cycle$VerticalLine@10385c1 linev (2,7) act: 0.0
Cycle$VerticalLine@42719c linev (16,3) act: 0.0
Cycle$DiagonalLineSWNE@30c221 diagswne (3,15) act: 0.0
Cycle$DiagonalLineSWNE@119298d diagswne (11,7) act: 0.0
Cycle$DiagonalLineSWNE@f72617 diagswne (3,6) act: 0.0
Cycle$DiagonalLineSWNE@1e5e2c3 diagswne (12,15) act: 0.0
Cycle$CornerSE@18a992f cornerse (2,7) act: 0.1
Cycle$CornerNE@4f1d0d cornerne (2,16) act: 0.1
Cycle$CornerSE@1fc4bec cornerse (3,5) act: 0.0
Cycle$CornerSE@dc8569 cornerse (4,4) act: 0.0
Cycle$CornerSE@1bab50a cornerse (4,13) act: 0.0
Cycle$CornerSE@c3c749 cornerse (5,3) act: 0.0
Cycle$CornerNW@6e1408 cornernw (5,5) act: 0.0
Cycle$CornerSE@e53108 cornerse (5,12) act: 0.0
Cycle$CornerNW@f62373 cornernw (5,14) act: 0.0
Cycle$CornerNW@19189e1 cornernw (6,13) act: 0.0
Cycle$CornerSE@1f33675 cornerse (7,2) act: 0.1
Cycle$CornerNE@7c6768 cornerne (7,11) act: 0.1
Cycle$CornerSW@1690726 cornersw (11,7) act: 0.1
Cycle$CornerNW@5483cd cornernw (11,16) act: 0.1
Cycle$CornerSE@9931f5 cornerse (12,5) act: 0.0
Cycle$CornerSE@19ee1ac cornerse (13,4) act: 0.0
Cycle$CornerNW@1f1fba0 cornernw (13,6) act: 0.0
Cycle$CornerSE@1befab0 cornerse (13,13) act: 0.0
Cycle$CornerNW@13c5982 cornernw (13,15) act: 0.0
Cycle$CornerNW@1186fab cornernw (14,5) act: 0.0
Cycle$CornerNW@14b7453 cornernw (14,14) act: 0.0
Cycle$CornerNW@c21495 cornernw (15,13) act: 0.0
Cycle$CornerSW@1d5550d cornersw (16,2) act: 0.1
Cycle$CornerNW@c2ea3f cornernw (16,11) act: 0.1
Cycle$CornerSW3D@a0dcd9 cornersw3d (11,7) act: 0.2
Cycle$CornerNE3D@1034bb5 cornerne3d (7,11) act: 0.0
Cycle$Square@15f5897 square (2,16) act: 1.1
Cycle$Square@b162d5 square (7,11) act: 0.9
Cycle$HorizontalLine@723d7c lineh (3,7) act: 0.30000000000000004
Cycle$HorizontalLine@22c95b lineh (3,16) act: 0.1
Cycle$VerticalLine@1d1acd3 linev (11,8) act: 0.30000000000000004
Cycle$VerticalLine@a981ca linev (2,8) act: 0.1
Cycle$HorizontalLine@8814e9 lineh (8,2) act: 0.1
Cycle$HorizontalLine@1503a3 lineh (8,11) act: 0.1
Cycle$VerticalLine@1a1c887 linev (16,3) act: 0.1
Cycle$VerticalLine@743399 linev (7,3) act: 0.1
Cycle$DiagonalLineSWNE@1d9dc39 diagswne (12,15) act: 0.4
Cycle$DiagonalLineSWNE@93dcd diagswne (3,6) act: 0.4
master concept list [Cycle$CubeA@1cfb549, Cycle$CubeB@87816d]
Cycle$CubeA@1cfb549 cubeA (7,11) act: 0.0 off =false
Cycle$CubeB@87816d cubeB (2,16) act: 0.0 off =true
active concepts are [Cycle$CubeB@87816d]
Cycle$CubeB@87816d cubeB (2,16) act: 0.0
~~~~~~~~~~~~~~~~~~~~~~~~
```

Appendix H. ApparentMotion.java Output

```
processing with horizontal priming
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
after detecting new paths, path list becomes
before top-down spreading activation
 e act: 0.5 (3,11)-(7,11)
 w act: 0.5 (15,7)-(11,7)
after top-down spreading activation
 e act: 0.5 (3,11)-(7,11)
 w act: 0.5 (15,7)-(11,7)
at detect paths, with existing path list
after detecting new paths, path list becomes
 e y=11
 w y=7
 e y=11
 w y=7
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.5 (7,11)-(11,11)
 w act: 0.5 (11,7)-(7,7)
 s act: 0.5 (11,7)-(11,11)
path reinforces segment e at y=11
path reinforces segment w at y=7
after top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.7 (7,11)-(11,11)
 w act: 0.7 (11,7)-(7,7)
 s act: 0.5 (11,7)-(11,11)
same origin for n,e
n: 0.5
e: 0.7
removed n from active list
same origin for w,s
w: 0.7
s: 0.5
removed s from active list
at detect paths, with existing path list
 e y=11
 w y=7
after detecting new paths, path list becomes
 e y=11
 w y=7
motion perceived with horizontal priming:
 e act: 0.7 (7,11)-(11,11)
 w act: 0.7 (11,7)-(7,7)
---------------
processing with vertical priming
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
after detecting new paths, path list becomes
before top-down spreading activation
 n act: 0.5 (7,15)-(7,11)
 s act: 0.5 (11,3)-(11,7)
after top-down spreading activation
 n act: 0.5 (7,15)-(7,11)
 s act: 0.5 (11,3)-(11,7)
at detect paths, with existing path list
after detecting new paths, path list becomes
 n x=7
 s x=11
 n x=7
 s x=11
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.5 (7,11)-(11,11)
 w act: 0.5 (11,7)-(7,7)
 s act: 0.5 (11,7)-(11,11)
path reinforces segment n at x=7
path reinforces segment s at x=11
after top-down spreading activation
 n act: 0.7 (7,11)-(7,7)
 e act: 0.5 (7,11)-(11,11)
 w act: 0.5 (11,7)-(7,7)
 s act: 0.7 (11,7)-(11,11)
same origin for n,e
n: 0.7
e: 0.5
removed e from active list
same origin for w,s
w: 0.5
s: 0.7
removed w from active list
at detect paths, with existing path list
 n x=7
 s x=11
after detecting new paths, path list becomes
 n x=7
 s x=11
```

```
motion perceived with vertical priming:
  n act: 0.7 (7,11)-(7,7)
  s act: 0.7 (11,7)-(11,11)
---------------
processing tall input
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
after detecting new paths, path list becomes
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.6666666666666666 (7,11)-(10,11)
  w act: 0.6666666666666666 (10,7)-(7,7)
  s act: 0.5 (10,7)-(10,11)
after top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.6666666666666666 (7,11)-(10,11)
  w act: 0.6666666666666666 (10,7)-(7,7)
  s act: 0.5 (10,7)-(10,11)
same origin for n,e
n: 0.5
e: 0.6666666666666666
removed n from active list
same origin for w,s
w: 0.6666666666666666
s: 0.5
removed s from active list
at detect paths, with existing path list
after detecting new paths, path list becomes
  e y=11
  w y=7
motion perceived with tall input:
  e act: 0.6666666666666666 (7,11)-(10,11)
  w act: 0.6666666666666666 (10,7)-(7,7)
---------------
processing wide input
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
after detecting new paths, path list becomes
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.4 (7,11)-(12,11)
  w act: 0.4 (12,7)-(7,7)
  s act: 0.5 (12,7)-(12,11)
after top-down spreading activation
```

```
 n act: 0.5 (7,11)-(7,7)
 e act: 0.4 (7,11)-(12,11)
 w act: 0.4 (12,7)-(7,7)
 s act: 0.5 (12,7)-(12,11)
same origin for n,e
n: 0.5
e: 0.4
removed e from active list
same origin for w,s
w: 0.4
s: 0.5
removed w from active list
at detect paths, with existing path list
after detecting new paths, path list becomes
 n x=7
 s x=12
motion perceived with wide input:
 n act: 0.5 (7,11)-(7,7)
 s act: 0.5 (12,7)-(12,11)
---------------
processing balanced input
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
after detecting new paths, path list becomes
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.5 (7,11)-(11,11)
 w act: 0.5 (11,7)-(7,7)
 s act: 0.5 (11,7)-(11,11)
after top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.5 (7,11)-(11,11)
 w act: 0.5 (11,7)-(7,7)
 s act: 0.5 (11,7)-(11,11)
same origin for n,e
n: 0.5
e: 0.5
tempi has same dest as another segment being considered
removing n because of w ending at 7,7
same origin for w,s
w: 0.5
s: 0.5
tempj has same dest as another segment being considered
removing s because of e ending at 11,11
at detect paths, with existing path list
```

```
after detecting new paths, path list becomes
  e y=11
  w y=7
motion perceived for balanced input:
  e act: 0.5 (7,11)-(11,11)
  w act: 0.5 (11,7)-(7,7)
---------------
processing increasing input
trial 1:  distance n,s = 4, distance e,w = 2
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
after detecting new paths, path list becomes
at decayPaths
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 1.0 (7,11)-(9,11)
  w act: 1.0 (9,7)-(7,7)
  s act: 0.5 (9,7)-(9,11)
after top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 1.0 (7,11)-(9,11)
  w act: 1.0 (9,7)-(7,7)
  s act: 0.5 (9,7)-(9,11)
same origin for n,e
n: 0.5
e: 1.0
removed n from active list
same origin for w,s
w: 1.0
s: 0.5
removed s from active list
at detect paths, with existing path list
after detecting new paths, path list becomes
  e y=11
  w y=7
motion perceived on trial 1:  distance n,s = 4, distance e,w = 2
  e act: 1.0 (7,11)-(9,11)
  w act: 1.0 (9,7)-(7,7)
- - - - - - - - - -
at decayPaths
  e y=11
  w y=7
path activation for e 0.5 decays to 0.3
path activation for w 0.5 decays to 0.3
  e y=11
```

```
 w y=7
trial 2:  distance n,s, = 4, distance e,w = 3
 e y=11
 w y=7
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
 e y=11
 w y=7
after detecting new paths, path list becomes
 e y=11
 w y=7
at decayPaths
 e y=11
 w y=7
path activation for e 0.3 decays to 0.09999999999999998
path activation for w 0.3 decays to 0.09999999999999998
 e y=11
 w y=7
 e y=11
 w y=7
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.6666666666666666 (7,11)-(10,11)
 w act: 0.6666666666666666 (10,7)-(7,7)
 s act: 0.5 (10,7)-(10,11)
path reinforces segment e at y=11
path reinforces segment w at y=7
after top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.8666666666666667 (7,11)-(10,11)
 w act: 0.8666666666666667 (10,7)-(7,7)
 s act: 0.5 (10,7)-(10,11)
same origin for n,e
n: 0.5
e: 0.8666666666666667
removed n from active list
same origin for w,s
w: 0.8666666666666667
s: 0.5
removed s from active list
at detect paths, with existing path list
 e y=11
 w y=7
after detecting new paths, path list becomes
 e y=11
```

```
  w y=7
motion perceived on trial 2:  distance n,s, = 4, distance e,w = 3
  e act: 0.8666666666666667 (7,11)-(10,11)
  w act: 0.8666666666666667 (10,7)-(7,7)
- - - - - - - - - -
at decayPaths
  e y=11
  w y=7
path activation for e 0.6 decays to 0.39999999999999997
path activation for w 0.6 decays to 0.39999999999999997
  e y=11
  w y=7
trial 3:  distance n,s, = 4, distance e,w = 4
  e y=11
  w y=7
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
  e y=11
  w y=7
after detecting new paths, path list becomes
  e y=11
  w y=7
at decayPaths
  e y=11
  w y=7
path activation for e 0.39999999999999997 decays to 0.19999999999999996
path activation for w 0.39999999999999997 decays to 0.19999999999999996
  e y=11
  w y=7
  e y=11
  w y=7
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.5 (7,11)-(11,11)
  w act: 0.5 (11,7)-(7,7)
  s act: 0.5 (11,7)-(11,11)
path reinforces segment e at y=11
path reinforces segment w at y=7
after top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.7 (7,11)-(11,11)
  w act: 0.7 (11,7)-(7,7)
  s act: 0.5 (11,7)-(11,11)
same origin for n,e
n: 0.5
```

```
e: 0.7
removed n from active list
same origin for w,s
w: 0.7
s: 0.5
removed s from active list
at detect paths, with existing path list
 e y=11
 w y=7
after detecting new paths, path list becomes
 e y=11
 w y=7
motion perceived on trial 3:  distance n,s, = 4, distance e,w = 4
 e act: 0.7 (7,11)-(11,11)
 w act: 0.7 (11,7)-(7,7)
----------
at decayPaths
 e y=11
 w y=7
path activation for e 0.7 decays to 0.49999999999999994
path activation for w 0.7 decays to 0.49999999999999994
 e y=11
 w y=7
trial 4:  distance n,s, = 4, distance e,w = 5
 e y=11
 w y=7
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
 e y=11
 w y=7
after detecting new paths, path list becomes
 e y=11
 w y=7
at decayPaths
 e y=11
 w y=7
path activation for e 0.49999999999999994 decays to 0.29999999999999993
path activation for w 0.49999999999999994 decays to 0.29999999999999993
 e y=11
 w y=7
 e y=11
 w y=7
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.4 (7,11)-(12,11)
```

```
  w act: 0.4 (12,7)-(7,7)
  s act: 0.5 (12,7)-(12,11)
 path reinforces segment e at y=11
 path reinforces segment w at y=7
 after top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.6000000000000001 (7,11)-(12,11)
  w act: 0.6000000000000001 (12,7)-(7,7)
  s act: 0.5 (12,7)-(12,11)
 same origin for n,e
 n: 0.5
 e: 0.6000000000000001
 removed n from active list
 same origin for w,s
 w: 0.6000000000000001
 s: 0.5
 removed s from active list
 at detect paths, with existing path list
  e y=11
  w y=7
 after detecting new paths, path list becomes
  e y=11
  w y=7
 motion perceived on trial 4:  distance n,s, = 4, distance e,w = 5
  e act: 0.6000000000000001 (7,11)-(12,11)
  w act: 0.6000000000000001 (12,7)-(7,7)
 ----------
 at decayPaths
  e y=11
  w y=7
 path activation for e 0.7999999999999999 decays to 0.5999999999999999
 path activation for w 0.7999999999999999 decays to 0.5999999999999999
  e y=11
  w y=7
 trial 5:  distance n,s, = 4, distance e,w = 6
  e y=11
  w y=7
 before top-down spreading activation
 after top-down spreading activation
 at detect paths, with existing path list
  e y=11
  w y=7
 after detecting new paths, path list becomes
  e y=11
  w y=7
 at decayPaths
```

```
  e y=11
  w y=7
path activation for e 0.5999999999999999 decays to 0.39999999999999986
path activation for w 0.5999999999999999 decays to 0.39999999999999986
  e y=11
  w y=7
  e y=11
  w y=7
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.3333333333333333 (7,11)-(13,11)
  w act: 0.3333333333333333 (13,7)-(7,7)
  s act: 0.5 (13,7)-(13,11)
path reinforces segment e at y=11
path reinforces segment w at y=7
after top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.5333333333333333 (7,11)-(13,11)
  w act: 0.5333333333333333 (13,7)-(7,7)
  s act: 0.5 (13,7)-(13,11)
same origin for n,e
n: 0.5
e: 0.5333333333333333
removed n from active list
same origin for w,s
w: 0.5333333333333333
s: 0.5
removed s from active list
at detect paths, with existing path list
  e y=11
  w y=7
after detecting new paths, path list becomes
  e y=11
  w y=7
motion perceived on trial 5:  distance n,s, = 4, distance e,w = 6
  e act: 0.5333333333333333 (7,11)-(13,11)
  w act: 0.5333333333333333 (13,7)-(7,7)
- - - - - - - - - -
at decayPaths
  e y=11
  w y=7
path activation for e 0.8999999999999999 decays to 0.7
path activation for w 0.8999999999999999 decays to 0.7
  e y=11
  w y=7
trial 6:  distance n,s, = 4, distance e,w = 7
```

```
 e y=11
 w y=7
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
 e y=11
 w y=7
after detecting new paths, path list becomes
 e y=11
 w y=7
at decayPaths
 e y=11
 w y=7
path activation for e 0.7 decays to 0.49999999999999994
path activation for w 0.7 decays to 0.49999999999999994
 e y=11
 w y=7
 e y=11
 w y=7
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.2857142857142857 (7,11)-(14,11)
 w act: 0.2857142857142857 (14,7)-(7,7)
 s act: 0.5 (14,7)-(14,11)
path reinforces segment e at y=11
path reinforces segment w at y=7
after top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.4857142857142857 (7,11)-(14,11)
 w act: 0.4857142857142857 (14,7)-(7,7)
 s act: 0.5 (14,7)-(14,11)
same origin for n,e
n: 0.5
e: 0.4857142857142857
removed e from active list
same origin for w,s
w: 0.4857142857142857
s: 0.5
removed w from active list
at detect paths, with existing path list
 e y=11
 w y=7
after detecting new paths, path list becomes
 e y=11
 w y=7
 n x=7
 s x=14
motion perceived on trial 6:  distance n,s, = 4, distance e,w = 7
 n act: 0.5 (7,11)-(7,7)
 s act: 0.5 (14,7)-(14,11)
- - - - - - - - - -
at decayPaths
 e y=11
 w y=7
 n x=7
 s x=14
path activation for e 0.49999999999999994 decays to 0.29999999999999993
path activation for w 0.49999999999999994 decays to 0.29999999999999993
path activation for n 0.5 decays to 0.3
path activation for s 0.5 decays to 0.3
 e y=11
 w y=7
 n x=7
 s x=14
trial 7:  distance n,s, = 4, distance e,w = 8
 e y=11
 w y=7
 n x=7
 s x=14
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
 e y=11
 w y=7
 n x=7
 s x=14
after detecting new paths, path list becomes
 e y=11
 w y=7
 n x=7
 s x=14
at decayPaths
 e y=11
 w y=7
 n x=7
 s x=14
path activation for e 0.29999999999999993 decays to 0.09999999999999992
path activation for w 0.29999999999999993 decays to 0.09999999999999992
path activation for n 0.3 decays to 0.09999999999999998
path activation for s 0.3 decays to 0.09999999999999998
 e y=11
 w y=7
```

```
  n x=7
  s x=14
  e y=11
  w y=7
  n x=7
  s x=14
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.25 (7,11)-(15,11)
  w act: 0.25 (15,7)-(7,7)
  s act: 0.5 (15,7)-(15,11)
path reinforces segment e at y=11
path reinforces segment w at y=7
path reinforces segment n at x=7
path reinforces segment s at x=15
after top-down spreading activation
  n act: 0.7 (7,11)-(7,7)
  e act: 0.45 (7,11)-(15,11)
  w act: 0.45 (15,7)-(7,7)
  s act: 0.7 (15,7)-(15,11)
same origin for n,e
n: 0.7
e: 0.45
removed e from active list
same origin for w,s
w: 0.45
s: 0.7
removed w from active list
at detect paths, with existing path list
  e y=11
  w y=7
  n x=7
  s x=14
after detecting new paths, path list becomes
  e y=11
  w y=7
  n x=7
  s x=15
motion perceived on trial 7:  distance n,s, = 4, distance e,w = 8
  n act: 0.7 (7,11)-(7,7)
  s act: 0.7 (15,7)-(15,11)
- - - - - - - - - -
at decayPaths
  e y=11
  w y=7
  n x=7
```

```
 s x=15
path activation for e 0.09999999999999992 decays to 0.0
path activation for w 0.09999999999999992 decays to 0.0
path activation for n 0.6 decays to 0.39999999999999997
path activation for s 0.6 decays to 0.39999999999999997
 n x=7
 s x=15
---------------
processing decreasing input
trial 1:  distance n,s, = 4, distance e,w = 8
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
after detecting new paths, path list becomes
at decayPaths
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.25 (7,11)-(15,11)
 w act: 0.25 (15,7)-(7,7)
 s act: 0.5 (15,7)-(15,11)
after top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.25 (7,11)-(15,11)
 w act: 0.25 (15,7)-(7,7)
 s act: 0.5 (15,7)-(15,11)
same origin for n,e
n: 0.5
e: 0.25
removed e from active list
same origin for w,s
w: 0.25
s: 0.5
removed w from active list
at detect paths, with existing path list
after detecting new paths, path list becomes
 n x=7
 s x=15
motion perceived on trial 1:  distance n,s, = 4, distance e,w = 8
 n act: 0.5 (7,11)-(7,7)
 s act: 0.5 (15,7)-(15,11)
- - - - - - - - - -
at decayPaths
 n x=7
 s x=15
path activation for n 0.5 decays to 0.3
path activation for s 0.5 decays to 0.3
```

```
  n x=7
  s x=15
 trial 2:  distance n,s, = 4, distance e,w = 7
  n x=7
  s x=15
 before top-down spreading activation
 after top-down spreading activation
 at detect paths, with existing path list
  n x=7
  s x=15
 after detecting new paths, path list becomes
  n x=7
  s x=15
 at decayPaths
  n x=7
  s x=15
 path activation for n 0.3 decays to 0.09999999999999998
 path activation for s 0.3 decays to 0.09999999999999998
  n x=7
  s x=15
  n x=7
  s x=15
 before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.2857142857142857 (7,11)-(14,11)
  w act: 0.2857142857142857 (14,7)-(7,7)
  s act: 0.5 (14,7)-(14,11)
 path reinforces segment n at x=7
 path reinforces segment s at x=14
 after top-down spreading activation
  n act: 0.7 (7,11)-(7,7)
  e act: 0.2857142857142857 (7,11)-(14,11)
  w act: 0.2857142857142857 (14,7)-(7,7)
  s act: 0.7 (14,7)-(14,11)
 same origin for n,e
 n: 0.7
 e: 0.2857142857142857
 removed e from active list
 same origin for w,s
 w: 0.2857142857142857
 s: 0.7
 removed w from active list
 at detect paths, with existing path list
  n x=7
  s x=15
 after detecting new paths, path list becomes
 n x=7
 s x=14
motion perceived on trial 2:  distance n,s, = 4, distance e,w = 7
 n act: 0.7 (7,11)-(7,7)
 s act: 0.7 (14,7)-(14,11)
----------
at decayPaths
 n x=7
 s x=14
path activation for n 0.6 decays to 0.39999999999999997
path activation for s 0.6 decays to 0.39999999999999997
 n x=7
 s x=14
trial 3:  distance n,s, = 4, distance e,w = 6
 n x=7
 s x=14
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
 n x=7
 s x=14
after detecting new paths, path list becomes
 n x=7
 s x=14
at decayPaths
 n x=7
 s x=14
path activation for n 0.39999999999999997 decays to 0.19999999999999996
path activation for s 0.39999999999999997 decays to 0.19999999999999996
 n x=7
 s x=14
 n x=7
 s x=14
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.3333333333333333 (7,11)-(13,11)
 w act: 0.3333333333333333 (13,7)-(7,7)
 s act: 0.5 (13,7)-(13,11)
path reinforces segment n at x=7
path reinforces segment s at x=13
after top-down spreading activation
 n act: 0.7 (7,11)-(7,7)
 e act: 0.3333333333333333 (7,11)-(13,11)
 w act: 0.3333333333333333 (13,7)-(7,7)
 s act: 0.7 (13,7)-(13,11)
same origin for n,e
```

```
n: 0.7
e: 0.3333333333333333
removed e from active list
same origin for w,s
w: 0.3333333333333333
s: 0.7
removed w from active list
at detect paths, with existing path list
 n x=7
 s x=14
after detecting new paths, path list becomes
 n x=7
 s x=13
motion perceived on trial 3:  distance n,s, = 4, distance e,w = 6
 n act: 0.7 (7,11)-(7,7)
 s act: 0.7 (13,7)-(13,11)
----------
at decayPaths
 n x=7
 s x=13
path activation for n 0.7 decays to 0.49999999999999994
path activation for s 0.7 decays to 0.49999999999999994
 n x=7
 s x=13
trial 4:  distance n,s, = 4, distance e,w = 5
 n x=7
 s x=13
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
 n x=7
 s x=13
after detecting new paths, path list becomes
 n x=7
 s x=13
at decayPaths
 n x=7
 s x=13
path activation for n 0.49999999999999994 decays to 0.29999999999999993
path activation for s 0.49999999999999994 decays to 0.29999999999999993
 n x=7
 s x=13
 n x=7
 s x=13
before top-down spreading activation
 n act: 0.5 (7,11)-(7,7)
 e act: 0.4 (7,11)-(12,11)
 w act: 0.4 (12,7)-(7,7)
 s act: 0.5 (12,7)-(12,11)
path reinforces segment n at x=7
path reinforces segment s at x=12
after top-down spreading activation
 n act: 0.7 (7,11)-(7,7)
 e act: 0.4 (7,11)-(12,11)
 w act: 0.4 (12,7)-(7,7)
 s act: 0.7 (12,7)-(12,11)
same origin for n,e
n: 0.7
e: 0.4
removed e from active list
same origin for w,s
w: 0.4
s: 0.7
removed w from active list
at detect paths, with existing path list
 n x=7
 s x=13
after detecting new paths, path list becomes
 n x=7
 s x=12
motion perceived on trial 4:  distance n,s, = 4, distance e,w = 5
 n act: 0.7 (7,11)-(7,7)
 s act: 0.7 (12,7)-(12,11)
- - - - - - - - - -
at decayPaths
 n x=7
 s x=12
path activation for n 0.7999999999999999 decays to 0.5999999999999999
path activation for s 0.7999999999999999 decays to 0.5999999999999999
 n x=7
 s x=12
trial 5:  distance n,s, = 4, distance e,w = 4
 n x=7
 s x=12
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
 n x=7
 s x=12
after detecting new paths, path list becomes
 n x=7
 s x=12
 at decayPaths
   n x=7
   s x=12
 path activation for n 0.5999999999999999 decays to 0.39999999999999986
 path activation for s 0.5999999999999999 decays to 0.39999999999999986
   n x=7
   s x=12
   n x=7
   s x=12
 before top-down spreading activation
   n act: 0.5 (7,11)-(7,7)
   e act: 0.5 (7,11)-(11,11)
   w act: 0.5 (11,7)-(7,7)
   s act: 0.5 (11,7)-(11,11)
 path reinforces segment n at x=7
 path reinforces segment s at x=11
 after top-down spreading activation
   n act: 0.7 (7,11)-(7,7)
   e act: 0.5 (7,11)-(11,11)
   w act: 0.5 (11,7)-(7,7)
   s act: 0.7 (11,7)-(11,11)
 same origin for n,e
 n: 0.7
 e: 0.5
 removed e from active list
 same origin for w,s
 w: 0.5
 s: 0.7
 removed w from active list
 at detect paths, with existing path list
   n x=7
   s x=12
 after detecting new paths, path list becomes
   n x=7
   s x=11
 motion perceived on trial 5:  distance n,s, = 4, distance e,w = 4
   n act: 0.7 (7,11)-(7,7)
   s act: 0.7 (11,7)-(11,11)
 - - - - - - - - - -
 at decayPaths
   n x=7
   s x=11
 path activation for n 0.8999999999999999 decays to 0.7
 path activation for s 0.8999999999999999 decays to 0.7
   n x=7
   s x=11
```

```
trial 6:  distance n,s, = 4, distance e,w = 3
  n x=7
  s x=11
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
  n x=7
  s x=11
after detecting new paths, path list becomes
  n x=7
  s x=11
at decayPaths
  n x=7
  s x=11
path activation for n 0.7 decays to 0.49999999999999994
path activation for s 0.7 decays to 0.49999999999999994
  n x=7
  s x=11
  n x=7
  s x=11
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 0.6666666666666666 (7,11)-(10,11)
  w act: 0.6666666666666666 (10,7)-(7,7)
  s act: 0.5 (10,7)-(10,11)
path reinforces segment n at x=7
path reinforces segment s at x=10
after top-down spreading activation
  n act: 0.7 (7,11)-(7,7)
  e act: 0.6666666666666666 (7,11)-(10,11)
  w act: 0.6666666666666666 (10,7)-(7,7)
  s act: 0.7 (10,7)-(10,11)
same origin for n,e
n: 0.7
e: 0.6666666666666666
removed e from active list
same origin for w,s
w: 0.6666666666666666
s: 0.7
removed w from active list
at detect paths, with existing path list
  n x=7
  s x=11
after detecting new paths, path list becomes
  n x=7
  s x=10
```

```
motion perceived on trial 6:  distance n,s, = 4, distance e,w = 3
  n act: 0.7 (7,11)-(7,7)
  s act: 0.7 (10,7)-(10,11)
- - - - - - - - - -
at decayPaths
  n x=7
  s x=10
path activation for n 1.0 decays to 0.8
path activation for s 1.0 decays to 0.8
  n x=7
  s x=10
trial 7:  distance n,s, = 4, distance e,w = 2
  n x=7
  s x=10
before top-down spreading activation
after top-down spreading activation
at detect paths, with existing path list
  n x=7
  s x=10
after detecting new paths, path list becomes
  n x=7
  s x=10
at decayPaths
  n x=7
  s x=10
path activation for n 0.8 decays to 0.6000000000000001
path activation for s 0.8 decays to 0.6000000000000001
  n x=7
  s x=10
  n x=7
  s x=10
before top-down spreading activation
  n act: 0.5 (7,11)-(7,7)
  e act: 1.0 (7,11)-(9,11)
  w act: 1.0 (9,7)-(7,7)
  s act: 0.5 (9,7)-(9,11)
path reinforces segment n at x=7
path reinforces segment s at x=9
after top-down spreading activation
  n act: 0.7 (7,11)-(7,7)
  e act: 1.0 (7,11)-(9,11)
  w act: 1.0 (9,7)-(7,7)
  s act: 0.7 (9,7)-(9,11)
same origin for n,e
n: 0.7
e: 1.0
```

```
removed n from active list
same origin for w,s
w: 1.0
s: 0.7
removed s from active list
at detect paths, with existing path list
 n x=7
 s x=10
after detecting new paths, path list becomes
 n x=7
 s x=10
 e y=11
 w y=7
motion perceived on trial 7:  distance n,s, = 4, distance e,w = 2
 e act: 1.0 (7,11)-(9,11)
 w act: 1.0 (9,7)-(7,7)
```

Appendix I. Domain.java Output Showing Self-Organization

```
at domain constructor
at initiateDomainAlpha
line is b,c
line is b,c
line is b,c
line is l,m,n,o,p
line is l,m,n
line is o,p
output list is [[b, c], [b, c], [b, c], [l, m, n, o, p], [l, m, n], [o,
p]]
getData is [[b, c], [b, c], [b, c], [l, m, n, o, p], [l, m, n], [o, p]]
time 1.0
inputData is
[b, c]
input is b
input b matches node for b
activation becomes 0.5
input is c
input c matches node for c
activation becomes 0.5
targets of spreading []

short list []

after adding new firing nodes, []

removing duplicates []

time 2.0
inputData is
[b, c]
input is b
input b matches node for b
activation becomes 1.0
input is c
input c matches node for c
activation becomes 1.0
** saw b
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 2.0
threshold is now 0.9
** saw c
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 2.0
threshold is now 0.9
spreading from b to a
activation of a is now 0.25
targets of spreading [Domain$Node@360be0]
 a
short list [Domain$Node@360be0]
```

```
 a
after adding new firing nodes, [Domain$Node@45a877,
Domain$Node@1372a1a]
 b c
removing duplicates [Domain$Node@45a877, Domain$Node@1372a1a]
 b c
creating links between co-firing nodes

 b c
creating link from b to c
time is 2.0
nodes firing at time 2.0
 [Domain$Node@45a877, Domain$Node@1372a1a]
 b c
using b as representative node
checking for gravitational effects on node b
old domain state []

new domain state [Domain$Collocation@f6a746]
 2.0
this node fires at time 2.0
in domain state, there is a collocation at time 2.0
distance from current node is 0.0
no positive collocation, so no positive gravitation
no negative collocation, so no negative gravitation
total gravitation is 0.0

time 3.0
inputData is
[b, c]
input is b
input b matches node for b
activation becomes 0.5
input is c
input c matches node for c
activation becomes 0.5
targets of spreading []

short list []

after adding new firing nodes, []

removing duplicates []

time 4.0
inputData is
[l, m, n, o, p]
input is l
input l matches node for l
activation becomes 0.5
input is m
input m matches node for m
activation becomes 0.5
input is n
input n matches node for n
activation becomes 0.5
input is o
input o matches node for o
activation becomes 0.5
input is p
input p matches node for p
activation becomes 0.5
targets of spreading []

short list []

after adding new firing nodes, []

removing duplicates []

time 5.0
time reset to 0.0
old domain state [Domain$Collocation@f6a746]
 2.0
new domain state []

inputData is
[l, m, n]
input is l
input l matches node for l
activation becomes 1.0
input is m
input m matches node for m
activation becomes 1.0
input is n
input n matches node for n
activation becomes 1.0
node a may fire at will, since time has cycled to last firing time of
0.0
node d may fire at will, since time has cycled to last firing time of
0.0
node e may fire at will, since time has cycled to last firing time of
0.0
node l may fire at will, since time has cycled to last firing time of
0.0
node m may fire at will, since time has cycled to last firing time of
0.0
node n may fire at will, since time has cycled to last firing time of
0.0
node o may fire at will, since time has cycled to last firing time of
0.0
node p may fire at will, since time has cycled to last firing time of
0.0
** saw l
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 0.0
threshold is now 0.9
** saw m
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 0.0
```

```
threshold is now 0.9
** saw n
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 0.0
threshold is now 0.9
targets of spreading []

short list []

after adding new firing nodes, [Domain$Node@15ff48b,
Domain$Node@affc70, Domain$Node@1e63e3d]
 l m n
removing duplicates [Domain$Node@15ff48b, Domain$Node@affc70,
Domain$Node@1e63e3d]
 l m n
creating links between co-firing nodes

 l m n
creating link from l to m
creating link from l to n
creating link from m to n
time is 0.0
nodes firing at time 0.0
 [Domain$Node@15ff48b, Domain$Node@affc70, Domain$Node@1e63e3d]
 l m n
using l as representative node
checking for gravitational effects on node l
old domain state [Domain$Collocation@f6a746]
 2.0
new domain state [Domain$Collocation@1004901]
 0.0
this node fires at time 0.0
in domain state, there is a collocation at time 0.0
distance from current node is 0.0
in old domain state, there is a collocation at time 2.0
distance from current node is 2.0
using positive collocation firing at time 2.0
number of nodes in positive collocation 2.0
distance to positive collocation 2.0
nodeWeight 1.0
distanceWeight 1.0
positive gravitation 2.1
no negative collocation, so no negative gravitation
total gravitation is 2.1

slowed down node l to 0.2
slowed down node m to 0.2
slowed down node n to 0.2
time 1.0
inputData is
[o, p]
input is o
input o matches node for o
activation becomes 1.0
input is p
input p matches node for p
activation becomes 1.0
** saw o
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 1.0
threshold is now 0.9
** saw p
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 1.0
threshold is now 0.9
targets of spreading []

short list []

after adding new firing nodes, [Domain$Node@1b90b39,
Domain$Node@18fe7c3]
 o p
removing duplicates [Domain$Node@1b90b39, Domain$Node@18fe7c3]
 o p
creating links between co-firing nodes

 o p
creating link from o to p
time is 1.0
nodes firing at time 1.0
 [Domain$Node@1b90b39, Domain$Node@18fe7c3]
 o p
using o as representative node
checking for gravitational effects on node o
old domain state [Domain$Collocation@f6a746]
 2.0
new domain state [Domain$Collocation@1004901,
Domain$Collocation@b8df17]
 0.0 1.0
this node fires at time 1.0
in domain state, there is a collocation at time 0.0
distance from current node is -1.0
in domain state, there is a collocation at time 1.0
distance from current node is 0.0
in old domain state, there is a collocation at time 2.0
distance from current node is 1.0
using positive collocation firing at time 2.0
number of nodes in positive collocation 2.0
distance to positive collocation 1.0
nodeWeight 1.0
distanceWeight 1.0
positive gravitation 3.1
using negative collocation firing at time 0.0
number of nodes in negative collocation 3.0
distance to negative collocation 1.0
nodeWeight 1.0
distanceWeight 1.0
negative gravitation -4.1
```

```
total gravitation is -0.9999999999999996

sped up node o to 0.8
sped up node p to 0.8
```

Appendix J. Cycle.java

```
//Cycle.java
//Katherine L. McCreight
// 0-15-04
//
// program detects figures in bitmap input, using features, objects, and concepts
// difference in decay rates creates object persistence and regularization of input
// attention to specific objects evokes switching behavior in perception of Necker cube

import java.io.*;
import java.util.StringTokenizer;
import java.util.*;
import java.lang.*;
import java.text.*;
import java.lang.Math.*;
import java.util.Arrays;

public class Cycle {
   public LinkedList activeConcepts = new LinkedList();
   public LinkedList masterConceptList = new LinkedList();
   public double conceptThreshold;
   public double conceptImplementationThreshold;
   public int conceptDelta;
   public double conceptTime;
   public double conceptTimePassed;
   public double conceptDecayAmount;
   public double conceptDecayFloor;
   public double initialConceptActivationValue;
   public double initialConceptThresholdValue;
   public double conceptActivationWeight;
   public LinkedList activeFeatures = new LinkedList();
   public LinkedList activeObjects = new LinkedList();
   //input is stored as a list of 400 elements, understood as 20 rows of 20 entries each
   // actual location of input is coded in the x,y values
   public LinkedList inputList = new LinkedList();
   public int objectDelta;
   public int featureDelta;
   // timing cycles
   public double featureTime;
   public double objectTime;
   public double oldTime;
   public double featureTimePassed;
   public double objectTimePassed;
   // proximity constraints on collocation
```

```
public double xlimit;
public double ylimit;
public double xylimit;
public double featureThreshold;
public double linkInitialValue;
public double actInitialValue;
public double threshInitialValue;
public double objectActivationWeight;
public double objectBasedActivationIncrement;
public double nodeWeight;
public double distanceWeight;
public double thresholdReturnRate;
// firing takes an interval of time, during which spreading occurs
// spreadDelay should be < firingInterval
// activation is reset at end of firing interval
public double firingInterval;
public double spreadDelay;
public double spreadAmount;
public double timeIncrement;
public double strengthIncrement;
public double assimilationAmount;
public double lateralObjectActivationIncrement;
public double conceptToObjectActivationIncrement;
public int postulationThreshold;
public double objectThreshold;
// when sending top-down activation, don't create features outside of screen area
int screenxmin;
int screenxmax;
int screenymin;
int screenymax;
public double featureInputWeight;
public double featureObjectWeight;
// implementation thresholds ignore low-activation features and objects
// in order to simplify calculations
public double featureImplementationThreshold;
public double objectImplementationThreshold;
public double overlayThreshold;
public LinkedList masterObjectList = new LinkedList();
public LinkedList horizontalLineObjectList = new LinkedList();
public LinkedList verticalLineObjectList = new LinkedList();
public LinkedList diagonalSWNELineObjectList = new LinkedList();
public LinkedList diagonalNWSELineObjectList = new LinkedList();
public LinkedList square3List = new LinkedList();
// decay
// keep floor above decay amount to avoid negative activation strengths
public double featureDecayAmount;
```

```
    public double featureDecayFloor;
    public double objectDecayAmount;
    public double objectDecayFloor;

    // white = 1, black = 0
    int[] v = {1, 0, 1, 1, 0, 1, 1, 0, 1};
    int[] h = {1, 1, 1, 0, 0, 0, 1, 1, 1};
    int[] i = {1, 0, 1, 0, 0, 0, 1, 0, 1};
    int[] ie = {1, 0, 1, 1, 0, 0, 1, 0, 1};
    int[] iw = {1, 0, 1, 0, 0, 1, 1, 0, 1};
    int[] in = {1, 0, 1, 0, 0, 0, 1, 1, 1};
    int[] is = {1, 1, 1, 0, 0, 0, 1, 0, 1};
    int[] cne = {1, 0, 1, 1, 0, 0, 1, 1, 1};
    int[] cse = {1, 1, 1, 1, 0, 0, 1, 0, 1};
    int[] csw = {1, 1, 1, 0, 0, 1, 1, 0, 1};
    int[] cnw = {1, 0, 1, 0, 0, 1, 1, 1, 1};
    // diagonal elements
    int[] cnedotsw= {1, 0, 1, 1, 0, 0, 0, 1, 1};
    int[] cnedotne = {1, 0, 0, 1, 0, 0, 1, 1, 1};
    int[] cnedotse = {1, 0, 1, 1, 0, 0, 1, 1, 0};
    int[] cnedotnw = {0, 0, 1, 1, 0, 0, 1, 1, 1};

    int[] cswdotsw = {1, 1, 1, 0, 0, 1, 0, 0, 1};
    int[] cswdotne = {1, 1, 0, 0, 0, 1, 1, 0, 1};
    int[] cswdotse = {1, 1, 1, 0, 0, 1, 1, 0, 0};
    int[] cswdotnw = {0, 1, 1, 0, 0, 1, 1, 0, 1};

    int[] csedotsw = {1, 1, 1, 1, 0, 0, 0, 0, 1};
    int[] csedotne = {1, 1, 0, 1, 0, 0, 1, 0, 1};
    int[] csedotse = {1, 1, 1, 1, 0, 0, 1, 0, 0};
    int[] csedotnw = {0, 1, 1, 1, 0, 0, 1, 0, 1};

    int[] cnwdotsw = {1, 0, 1, 0, 0, 1, 0, 1, 1};
    int[] cnwdotne = {1, 0, 0, 0, 0, 1, 1, 1, 1};
    int[] cnwdotse = {1, 0, 1, 0, 0, 1, 1, 1, 0};
    int[] cnwdotnw = {0, 0, 1, 0, 0, 1, 1, 1, 1};

    int[] diagnwse = {0, 1, 1, 1, 0, 1, 1, 1, 0};
    int[] diagswne = {1, 1, 0, 1, 0, 1, 0, 1, 1};

    int[] dot = {1, 1, 1, 1, 0, 1, 1, 1, 1};

....

// make a matrix so each BitGrid is stored in the matrix location
    // corresponding to its input location
```

```
   public BitGrid[][] bitGridMatrix = new BitGrid[20][20];
   //  make a matrix so each featureList is stored in the matrix location
   // corresponding to its input location
   public FeatureDetector[][][] featureListMatrix = new FeatureDetector[20][20][40];

   public FeatureDetector[] featureList1$1 = new FeatureDetector[40];
   public FeatureDetector[] featureList1$2 = new FeatureDetector[40];
   public FeatureDetector[] featureList1$3 = new FeatureDetector[40];
   public FeatureDetector[] featureList1$4 = new FeatureDetector[40];
   public FeatureDetector[] featureList1$5 = new FeatureDetector[40];
   public FeatureDetector[] featureList1$6 = new FeatureDetector[40];
   public FeatureDetector[] featureList1$7 = new FeatureDetector[40];
   public FeatureDetector[] featureList1$8 = new FeatureDetector[40];

 ....
//------------- sub classes -----------------
   class BitGrid  {
         int x;
         int y;
         int[] input = new int[9];

         // constructor
         BitGrid(int xloc, int yloc) {
            this.x = xloc;
            this.y = yloc;
         } // end constructor
   } // end class BitGrid
   // -------------
   class FeatureDetector  {
         String description;
         int x;
         int y;
         LinkedList neighbors = new LinkedList();
         LinkedList links = new LinkedList();
         double act;
         double baseThresh;
         double currentThresh;
         double bitAct;
         double objectAct;
         int[] inputCode = new int[9];
         boolean off;
         double lastTime;
         double adjustedTime;

         public void getAct(double time) {
            this.act = this.bitAct + this.objectAct;
} // end method FeatureDetector.getAct

public void checkInhibition(double time) {
   if (time>=(adjustedTime+featureDelta)) {
         this.off = false;
   } // end if enough time has passed
} // end method FeatureDetector.checkInhibition

public void fire(double time) {
   this.currentThresh = currentThresh - 0.1*currentThresh;
   this.lastTime = time;
   this.adjustedTime = this.lastTime;
   this.off = true;
   activeFeatures.add(this);
   this.act = 0.0;
} // end method FeatureDetector.fire

// constructor
FeatureDetector(int[] v, int x, int y, String d) {
   this.x = x;
   this.y = y;
   this.inputCode = v;
   this.act = actInitialValue;
   this.baseThresh = threshInitialValue;
   this.currentThresh = baseThresh;
   this.off = false;
   this.description = d;
} // end constructor

FeatureDetector(int[] v, int x, int y) {
   this.x = x;
   this.y = y;
   this.inputCode = v;
   this.act = actInitialValue;
   this.baseThresh = threshInitialValue;
   this.currentThresh = baseThresh;
   this.off = false;
} // end constructor

FeatureDetector(int[] v) {
   inputCode = v;
   act = actInitialValue;
   baseThresh = threshInitialValue;
   currentThresh = baseThresh;
   off = false;
} // end constructor
```

```
	FeatureDetector() {
	   act = actInitialValue;
	   baseThresh = threshInitialValue;
	   currentThresh = baseThresh;
	   off = false;
	} // end constructor

} // end class FeatureDetector
// -------------

class Concept {
	String name;
	int x;
	int y;
	int size;
	LinkedList relatedObjects;
	// code for firing and inhibition
	double act;
	double baseThresh;
	double currentThresh;
	boolean off;
	double lastTime;
	double adjustedTime;

	public void checkInhibition(double time) {
	   if (time>=(adjustedTime+conceptDelta)) {
		this.off = false;
	   } // end if enough time has passed
	} // end concept.checkInhibition

	public void fire(double time) {
	   this.currentThresh = currentThresh - 0.1*currentThresh;
	   this.lastTime = time;
	   this.adjustedTime = this.lastTime;
	   this.off = true;
	   activeConcepts.add(this);
	   this.act = 0.0;
	} // end method concept.fire

	// default constructor
	Concept() {
	   this.act = initialConceptActivationValue;
	   this.baseThresh = initialConceptThresholdValue;
	   this.currentThresh = baseThresh;
	   this.off = false;
```

```
        } // end constructor

        Concept(String n) {
           this.name = n;
           this.act = initialConceptActivationValue;
           this.baseThresh = initialConceptThresholdValue;
           this.currentThresh = baseThresh;
           this.off = false;
        } // end constructor

        Concept(String n, int xloc, int yloc) {
           this.name = n;
           this.x = xloc;
           this.y = yloc;

           this.act = initialConceptActivationValue;
           this.baseThresh = initialConceptThresholdValue;
           this.currentThresh = baseThresh;
           this.off = false;
        } // end constructor

} // end class Concept
//----------
class HorizontalLine extends Percept{
        int xmin;
        int xmax;

        // method to derive related Feature Detectors (for top-down activation)
        // screen out any short lines <3
        // variables x and y are inherited from percept
        public LinkedList getFD() {
           LinkedList fd = new LinkedList();
           for (int i=xmin+1; i<xmax; i++){
                 //FeatureDetector fdi = get fd for y=this.y, x=i, 1=h
                 if ((i<=screenxmax)&(i>=screenxmin)) {
                    FeatureDetector fdi = featureListMatrix[i][y][1];
                    fd.add(fdi);
                 } // end if in screen area
           } // end for i center locations along x axis
           return fd;
        } // end method HorizontalLine.getFD

        //default constructor
        HorizontalLine() {
           type = "lineh";
        }
```

```
// by convention, set percept value x to xmin
// so that percept can have both x and y values specified

// constructor for one feature h
HorizontalLine(FeatureDetector f) {
   xmin = f.x - 1;  // adjust for width of bit grid
   xmax = f.x + 1;
   y = f.y;
   x = xmin;  // percept level value
   size = xmax - xmin;
   type = "lineh";
} // end constructor

// constructor for two features h
// should only be called when f1.y = f2.y and |f1.x - f2.x | < range r
// note features are located on centers
HorizontalLine(FeatureDetector f1, FeatureDetector f2) {
   int f1x = f1.x;
   int f2x = f2.x;
   if (f1x<f2x) {
         this.xmin = f1.x - 1;
         this.xmax = f2.x + 1; // add or subtract 1 for width of bit grid
   } // assign f1 as left
   if (f1x>f2x) {
         this.xmin = f2.x - 1;
         this.xmax = f1.x + 1; // add or subtract 1 for width of bit grid
   } // assign f2 as left
   this.y = f1.y;
   this.x = xmin; // percept level value
   this.size = xmax -xmin;
   type = "lineh";
} // end constructor

// constructor given two HorizontalLine objects
// should only be called when h1.y = h2.y,
// and, assuming h1.x < h2.x, h2.xmin - h1.xmax < range r
HorizontalLine(HorizontalLine h1, HorizontalLine h2) {
   if (h1.xmin < h2.xmin) {
         this.xmin = h1.xmin;
         this.xmax = h2.xmax;
   }  // end if h1 is left of h2
   if (h1.xmin > h2.xmin) {
         this.xmin = h2.xmin;
         this.xmax = h1.xmax;
   }  // end if h1 is left of h2
```

```
            this.y = h1.y;
            this.x = xmin; // percept level value
            this.size = xmax - xmin;
            type = "lineh";
          } // end constructor

          // constructor given one feature detector h and one HorizontalLine object
          // should only be called when fd.y = HL.y, and distance between their ends is <
range r
          HorizontalLine(HorizontalLine h, FeatureDetector f) {
            if(h.xmin < f.x) {
                  this.xmin = h.xmin;
                  this.xmax = f.x + 1; // add 1 for width of bitgrid
            } // end if h is left of f
            if (h.xmin > f.x) {
                  this.xmin = f.x - 1;  // subtract 1 fpr width of bitgrid
                  this.xmax = h.xmax;
            }
            this.y = h.y;
            this.x = xmin;  // percept level value
            this.size = xmax - xmin;
            type = "lineh";
          } // end constructor
    } // end class HorizontalLine

    //----------
    class VerticalLine extends Percept{
          int ymin;
          int ymax;

          // method to derive related Feature Detectors (for top-down activation)
          public LinkedList getFD() {
            LinkedList fd = new LinkedList();
            for (int i=ymin+1; i<ymax; i++){
                  //FeatureDetector fdi = get fd for y=this.x, y=i, 0=v
                  if ((i<=screenymax)&(i>=screenymin)) {
                    FeatureDetector fdi = featureListMatrix[x][i][0];
                    fd.add(fdi);
                  } // end if in screen area
            } // end for i center locations along y axis
            return fd;
          } // end method VerticalLine.getFD

          //default constructor
          VerticalLine() {
            type = "linev";
```

```
}

// by convention, set percept value y to ymin
// so that percept can have both x and y values

// constructor for one feature v
VerticalLine(FeatureDetector f) {
   ymin = f.y - 1;  // adjust for width of bit grid
   ymax = f.y + 1;
   x = f.x;
   y = ymin;  // percept level value
   size = ymax - ymin;
   type = "linev";
} // end constructor

// constructor for two features v
// should only be called when f1.x = f2.x and |f1.y - f2.y | < range r
// note features are located on centers
VerticalLine(FeatureDetector f1, FeatureDetector f2) {
   int f1y = f1.y;
   int f2y = f2.y;
   if (f1y<f2y) {
         this.ymin = f1.y - 1;
         this.ymax = f2.y + 1; // add or subtract 1 for width of bit grid
   } // assign f1 as left
   if (f1y>f2y) {
         this.ymin = f2.y - 1;
         this.ymax = f1.y + 1; // add or subtract 1 for width of bit grid
   } // assign f2 as left
   this.x = f1.x;
   this.y = ymin; // percept level value
   this.size = ymax - ymin;
   type = "linev";
} // end constructor

// constructor given two VerticalLine objects
// should only be called when v1.x = v2.x,
// and, assuming v1.y < v2.y, v2.ymin - v1.ymax < range r
VerticalLine(VerticalLine v1, VerticalLine v2) {
   if (v1.ymin < v2.ymin) {
         this.ymin = v1.ymin;
         this.ymax = v2.ymax;
   }  // end if v1 is left of v2
   if (v1.ymin > v2.ymin) {
         this.ymin = v2.ymin;
         this.ymax = v1.ymax;
```

```
            } // end if v1 is left of v2
            this.x = v1.x;
            this.y = ymin; // percept level value
            this.size = ymax - ymin;
            type = "linev";
        } // end constructor

        // constructor given one feature detector h and one VerticalLine object
        // should only be called when fd.x = HL.x, and distance between their ends is <
range r
        VerticalLine(VerticalLine h, FeatureDetector f) {
            if(h.ymin < f.y) {
                this.ymin = h.ymin;
                this.ymax = f.y + 1; // add 1 for width of bitgrid
            } // end if h is left of f
            if (h.ymin > f.y) {
                this.ymin = f.y - 1;  // subtract 1 fpr width of bitgrid
                this.ymax = h.ymax;
            }
            this.x = h.x;
            this.y = ymin;  // percept level value
            this.size = ymax - ymin;
            type = "linev";
        } // end constructor

    } // end class VerticalLine
    //-----------------
    class DiagonalLineSWNE extends Percept {
        int xmin;
        int xmax;
        int ymin;
        int ymax;

        int xsw;
        int ysw;
        int xne;
        int yne;

        // method to derive related Feature Detectors (for top-down activation)
        // variables x and y are inherited form Percept
        public LinkedList getFD() {
            LinkedList fd = new LinkedList();
            // start with max y value, at sw endpoint
            // x value increases, y value decreases
            // note: feature detectors are measured on centers
            for (int i=xsw+1, j=ysw-1; i<xne-1; i++, j--) {
```

```
            if ((i<=screenxmax)&(i>=screenxmin)){
               FeatureDetector fdi = featureListMatrix[i][j][28];
               fd.add(fdi);
            } // end if in screen range
   } // end for i x
   return fd;
} // end method DiagonalLineSWNE.getFD

//default constructor
DiagonalLineSWNE() {
   type = "diagswne";
} // end default constructor

//by convention, set percept values for x and y to xsw and ysw

//constructor for one feature diagswne
DiagonalLineSWNE(FeatureDetector f) {
   //adjust for width of bit grid
   // moving from sw to ne, y decreases, x increases
   xmin = f.x - 1;
   ymax = f.y + 1;
   xmax = f.x + 1;
   ymin = f.y - 1;

   xsw = xmin;
   ysw = ymax;
   xne = xmax;
   yne = ymin;

   // set percept level values
   y = ysw;
   x = xsw;

   size = xne - xsw;
   type = "diagswne";
} // end constructor

//constructor for two features diagswne
// should only be called when |f1.x-f2.x|<range
// note features are located on centers, must adjust to get endpoints
// moving from sw to ne, x increases, y decreases
DiagonalLineSWNE(FeatureDetector f1, FeatureDetector f2) {
   int f1x = f1.x;
   int f2x = f2.x;
   if (f1x<f2x) {
         this.xmin = f1.x - 1;
```

```
            this.xmax = f2.x + 1;
            this.ymin = f2.y - 1;
            this.ymax = f1.y + 1;
        } // assign f1 as left
        if (f1x>f2x) {
            this.xmin = f2.x -1;
            this.xmax = f1.x +1;
            this.ymin = f1.y - 1;
            this.ymax = f2.y +1;
        } // assign f2 as left

        xsw = xmin;
        ysw = ymax;
        xne = xmax;
        yne = ymin;

        // percept values
        // use coordinates of sw corner -- requires ymax
        this.x = xmin;
        this.y = ymax;
        this.size = xmax - xmin;
        type = "diagswne";
    } // end constructor

    //constructor given two DiagonalLineSWNE objects
    // should only be called when, given d1.x<d2.x, d2.xmin-d1.xmax<range
    DiagonalLineSWNE(DiagonalLineSWNE d1, DiagonalLineSWNE d2) {
        if (d1.xmin < d2.xmin) {
            this.xmin = d1.xmin;
            this.xmax = d2.xmax;
            this.ymin = d2.ymin;
            this.ymax = d1.ymax;
        } // end if d1 is left of d2
        if (d1.xmin > d2.xmin) {
            this.xmin = d2.xmin;
            this.xmax = d1.xmax;
            this.ymin = d1.ymin;
            this.ymax = d2.ymax;
        } // end if d2 is left of d1

        xsw = xmin;
        ysw = ymax;
        xne = xmax;
        yne = ymin;

        // percept values
```

```
            this.x = xmin;
            this.y = ymax;
            this.size = xmax - xmin;
            type = "diagswne";
        } // end constructor

        // constructor given one feature detector h and one DiagonalLineSWNE object d
        // should only be called when distance between ends is < range
        DiagonalLineSWNE(DiagonalLineSWNE d, FeatureDetector f) {
            if (d.xmin < f.x) {
                    this.xmin = d.xmin;
                    this.xmax = f.x + 1;
                    this.ymin = f.y - 1;
                    this.ymax = d.ymax;
            } // end if d is left of f
            if (d.xmin > f.x) {
                    this.xmin = f.x - 1;
                    this.xmax = d.xmax;
                    this.ymin = d.ymin;
                    this.ymax = f.y + 1;
            } // end if f is left of d

            xsw = xmin;
            ysw = ymax;
            xne = xmax;
            yne = ymin;

            // percept values
            this.x = xmin;
            this.y = ymax;
            this.size = xmax - xmin;
            type = "diagswne";
        } // end constructor

    } // end class DiagonalLineSWNE
    //-----------------
    class DiagonalLineNWSE extends Percept {
        int xmin;
        int xmax;
        int ymin;
        int ymax;

        int xnw;
        int ynw;
        int xse;
        int yse;
```

```
// method to derive related Feature Detectors (for top-down activation)
// variables x and y are inherited form Percept
public LinkedList getFD() {
   LinkedList fd = new LinkedList();
   // start with xmin, ymin at nw endpoint
   // x value increases, y value increases
   for (int i=xnw,j=ynw; i<xse; i++,j++) {
         if ((i<=screenxmax)&(i>=screenxmin)){
            FeatureDetector fdi = featureListMatrix[i][j][27];
            fd.add(fdi);
         } // end if in screen range
   } // end for i x
   return fd;
} // end method DiagonalLineNWSE.getFD

//default constructor
DiagonalLineNWSE() {
   type = "diagnwse";
} // end default constructor

//by convention, set percept values for x and y to xsw and ysw

//constructor for one feature diagnwse
DiagonalLineNWSE(FeatureDetector f) {
   //adjust for width of bit grid
   // moving from nw to se, y increases, x increases

   xmin = f.x - 1;
   ymin = f.y - 1;
   xmax = f.x + 1;
   ymax = f.y + 1;

   xnw = xmin;
   ynw = ymin;
   xse = xmax;
   yse = ymax;

   // set percept level values
   y = ynw;
   x = xnw;
   size = xnw - xse;
   type = "diagnwse";
} // end constructor

//constructor for two features diagnwse
```

```
// should only be called when |f1.x-f2.x|<range
// note features are located on centers, must adjust to get endpoints
// moving from sw to ne, x increases, y decreases
DiagonalLineNWSE(FeatureDetector f1, FeatureDetector f2) {
   int f1x = f1.x;
   int f2x = f2.x;
   if (f1x<f2x) {
         this.xmin = f1.x - 1;
         this.xmax = f2.x + 1;
         this.ymin = f2.y - 1;
         this.ymax = f1.y + 1;
   } // assign f1 as left
   if (f1x>f2x) {
         this.xmin = f2.x - 1;
         this.xmax = f1.x + 1;
         this.ymin = f1.y - 1;
         this.ymax = f2.y + 1;
   } // assign f2 as left

   xnw = xmin;
   ynw = ymin;
   xse = xmax;
   yse = ymax;

   // percept values
   // use coordinates of nw corner
   this.x = xmin;
   this.y = ymin;
   this.size = xmax - xmin;
   type = "diagnwse";
} // end constructor

//constructor given two DiagonalLineNWSE objects
// should only be called when, given d1.x<d2.x, d2.xmin-d1.xmax<range
DiagonalLineNWSE(DiagonalLineNWSE d1, DiagonalLineNWSE d2) {
   if (d1.xmin < d2.xmin) {
         this.xmin = d1.xmin;
         this.xmax = d2.xmax;
         this.ymin = d2.ymin;
         this.ymax = d1.ymax;
   } // end if d1 is left of d2
   if (d1.xmin > d2.xmin) {
         this.xmin = d2.xmin;
         this.xmax = d1.xmax;
         this.ymin = d1.ymin;
         this.ymax = d2.ymax;
```

```
            } // end if d2 is left of d1

            xnw = xmin;
            ynw = ymin;
            xse = xmax;
            yse = ymax;

            // percept values
            this.x = xmin;
            this.y = ymin;
            this.size = xmax - xmin;
            type = "diagnwse";
        } // end constructor

        // constructor given one feature detector h and one DiagonalLineNWSE object d
        // should only be called when distance between ends is < range
        DiagonalLineNWSE(DiagonalLineNWSE d, FeatureDetector f) {
            if (d.xmin < f.x) {
                    this.xmin = d.xmin;
                    this.xmax = f.x + 1;
                    this.ymin = f.y - 1;
                    this.ymax = d.ymax;
            } // end if d is left of f
            if (d.xmin > f.x) {
                    this.xmin = f.x - 1;
                    this.xmax = d.xmax;
                    this.ymin = d.ymin;
                    this.ymax = f.y + 1;
            } // end if f is left of d

            xnw = xmin;
            ynw = ymin;
            xse = xmax;
            yse = ymax;

            // percept values
            this.x = xmin;
            this.y = ymin;
            this.size = xmax - xmin;
            type = "diagnwse";
        } // end constructor
    } // end class DiagonalLineNWSE
    // ------------------
    // class Percept is the super-class for Square, Square3, HorizontalLine, VerticalLine,
    // CornerNW, CornerSW, CornerNE, and CornerSE
    class Percept {
```

```
String type;
int x;
int y;
int size;
LinkedList parts;
// code for firing and inhibition
double act;
double baseThresh;
double currentThresh;
boolean off;
double lastTime;
double adjustedTime;

public void checkInhibition(double time) {
   if (time>=(adjustedTime+objectDelta)) {
         this.off = false;
   } // end if enough time has passed
} // end percept.checkInhibition

public void fire(double time) {
   this.currentThresh = currentThresh - 0.1*currentThresh;
   this.lastTime = time;
   this.adjustedTime = this.lastTime;
   this.off = true;
   activeObjects.add(this);
   this.act = 0.0;
} // end method percept.fire

// default constructor
Percept() {
   this.act = actInitialValue;
   this.baseThresh = threshInitialValue;
   this.currentThresh = baseThresh;
   this.off = false;
} // end constructor

Percept(String n) {
   this.type = n;
   this.act = actInitialValue;
   this.baseThresh = threshInitialValue;
   this.currentThresh = baseThresh;
   this.off = false;
} // end constructor

Percept(String n, int xloc, int yloc) {
   this.type = n;
```

```
            this.x = xloc;
            this.y = yloc;
            this.act = actInitialValue;
            this.baseThresh = threshInitialValue;
            this.currentThresh = baseThresh;
            this.off = false;
        } // end constructor
} // end class Percept

//-----------------
// class square inherits x, y, parts, type
// components are objects, built from features
// for example, square contains HorizontalLine, which contains h features
class Square extends Percept{
        int offset;
        String subtype;
        CornerNE scne;
        CornerSE scse;
        CornerSW scsw;
        CornerNW scnw;
        HorizontalLine shn;
        HorizontalLine shs;
        VerticalLine sve;
        VerticalLine svw;
        LinkedList componentObjects = new LinkedList();

        // method to derive related Feature Detectors (for top-down activation)
        public LinkedList getFD() {
          LinkedList fd = new LinkedList();
          LinkedList allDescendants = new LinkedList();
          for (int i=0; i<componentObjects.size(); i++) {
                Percept partsi = (Percept)componentObjects.get(i);
                LinkedList descendants = new LinkedList();
                if (partsi.type=="cornernw") {
                  CornerNW cnw = (CornerNW)partsi;
                  descendants = cnw.getFD();
                } else if (partsi.type=="cornersw") {
                  CornerSW csw = (CornerSW)partsi;
                  descendants = csw.getFD();
                } else if (partsi.type=="cornerne") {
                  CornerNE cne = (CornerNE)partsi;
                  descendants = cne.getFD();
                } else if (partsi.type=="cornerse") {
                  CornerSE cse = (CornerSE)partsi;
                  descendants = cse.getFD();
                } else if (partsi.type=="lineh") {
```

```
            HorizontalLine lh = (HorizontalLine)partsi;
            descendants = lh.getFD();
         } else if (partsi.type=="linev") {
            VerticalLine lv = (VerticalLine)partsi;
            descendants = lv.getFD();
         }// end if else parts
         allDescendants.addAll(descendants);
   } // end for i parts
   fd = allDescendants;
   return fd;
} // end method Square.getFD

//default constructor
Square() {
} // end default constructor

Square(String n) {
   this.type = "square";
   this.subtype = n;
} // end constructor

Square(int xloc, int yloc, int sizen) {
   this.type = "square";
   this.x = xloc;
   this.y = yloc;
   this.size = sizen;
} // end constructor

// have constructors for each pair of corners:
// ne/se, ne/nw, ne/sw, se/sw, se/nw, sw/nw

//CornerNE is anchor in lower left
Square(CornerNE cornerne, CornerSE cornerse) {
   type = "square";
   subtype = "generic";
   parts = new LinkedList();

   scne = cornerne;
   scse = cornerse;
   // get offset
   offset = scne.y - scse.y;
   size = offset; // percept-level variable

   // derive other objects
   scsw = new CornerSW();
   scnw = new CornerNW();
```

```
    // assign values
    scsw.x = scne.x + offset;
    scsw.y = scse.y;
    scnw.x = scsw.x;
    scnw.y = scne.y;
    // line objects
    shn = new HorizontalLine();
    shs = new HorizontalLine();
    svw = new VerticalLine();
    sve = new VerticalLine();
    // assign values
    shn.xmin = scse.x + 1;  // add 1 for partial width of corner object
    shn.xmax = scsw.x - 1;
    shn.x = shn.xmin; // by convention, so that Percept has x and y values
    shn.y = scse.y;
    shs.xmin = shn.xmin;
    shs.xmax = shn.xmax;
    shs.x = shs.xmin; // ditto
    shs.y = scne.y;
    sve.x = scsw.x;
    sve.ymin = scsw.y + 1;
    sve.ymax = scnw.y - 1;
    sve.y = sve.ymin; // ditto
    svw.x = scne.x;
    svw.ymin = sve.ymin;
    svw.ymax = sve.ymax;
    svw.y = svw.ymin; // ditto
    shn.size = shn.xmax - shn.xmin;
    shs.size = shs.xmax - shs.xmin;
    sve.size = sve.ymax - sve.ymin;
    svw.size = svw.ymax - svw.ymin;
    // percept level location variables
    this.x = scne.x;
    this.y = scne.y;
    //percept level parts variable
    componentObjects.add(scne);
    componentObjects.add(scnw);
    componentObjects.add(scse);
    componentObjects.add(scsw);
    componentObjects.add(shn);
    componentObjects.add(shs);
    componentObjects.add(sve);
    componentObjects.add(svw);
} // end constructor for two corners ne se

//CornerNE is anchor in lower left
```

```
Square(CornerNE cornerne, CornerNW cornernw) {
  type = "square";
  subtype = "generic";
  parts = new LinkedList();
  scne = cornerne;
  scnw = cornernw;
  // get offset
  offset = scnw.x - scne.x;
  size = offset; // percept-level variable
  // derive other objects
  scsw = new CornerSW();
  scse = new CornerSE();
  // assign values
  scsw.x = scnw.x;
  scsw.y = scnw.y - offset;
  scse.x = scne.x;
  scse.y = scne.y - offset;
  // line objects
  shn = new HorizontalLine();
  shs = new HorizontalLine();
  svw = new VerticalLine();
  sve = new VerticalLine();
  // assign values
  shn.xmin = scne.x + 1; // add 1 for partial width of corner object
  shn.xmax = scsw.x - 1;
  shn.x = shn.xmin; // by convention, so that Percept has x and y values
  shn.y = scse.y;
  shs.xmin = shn.xmin;
  shs.xmax = shn.xmax;
  shs.x = shs.xmin; // ditto
  shs.y = scne.y;
  sve.x = scsw.x;
  sve.ymin = scsw.y + 1;
  sve.ymax = scnw.y - 1;
  sve.y = sve.ymin; // ditto
  svw.x = scne.x;
  svw.ymin = sve.ymin;
  svw.ymax = sve.ymax;
  svw.y = svw.ymin; // ditto
  shn.size = shn.xmax - shn.xmin;
  shs.size = shs.xmax - shs.xmin;
  sve.size = sve.ymax - sve.ymin;
  svw.size = svw.ymax - svw.ymin;
  // percept level location variables
  this.x = scne.x;
  this.y = scne.y;
```

```
    //percept level parts variable
    componentObjects.add(scne);
    componentObjects.add(scnw);
    componentObjects.add(scse);
    componentObjects.add(scsw);
    componentObjects.add(shn);
    componentObjects.add(shs);
    componentObjects.add(sve);
    componentObjects.add(svw);
} // end constructor for two corners ne nw

//CornerNE is anchor in lower left
Square(CornerNE cornerne, CornerSW cornersw) {
    type = "square";
    subtype = "generic";
    parts = new LinkedList();
    scne = cornerne;
    scsw = cornersw;
    // get offset
    offset = scsw.x - scne.x;
    size = offset; // percept-level variable
    // derive other objects
    scnw = new CornerNW();
    scse = new CornerSE();
    // assign values
    scnw.x = scsw.x;
    scnw.y = scsw.y + offset;
    scse.x = scne.x;
    scse.y = scne.y - offset;
    // line objects
    shn = new HorizontalLine();
    shs = new HorizontalLine();
    svw = new VerticalLine();
    sve = new VerticalLine();
    // assign values
    shn.xmin = scne.x + 1; // add 1 for partial width of corner object
    shn.xmax = scsw.x - 1;
    shn.x = shn.xmin; // by convention, so that Percept has x and y values
    shn.y = scse.y;
    shs.xmin = shn.xmin;
    shs.xmax = shn.xmax;
    shs.x = shs.xmin; // ditto
    shs.y = scne.y;
    sve.x = scsw.x;
    sve.ymin = scsw.y + 1;
    sve.ymax = scnw.y - 1;
```

```
            sve.y = sve.ymin; // ditto
            svw.x = scne.x;
            svw.ymin = sve.ymin;
            svw.ymax = sve.ymax;
            svw.y = svw.ymin; // ditto
            shn.size = shn.xmax - shn.xmin;
            shs.size = shs.xmax - shs.xmin;
            sve.size = sve.ymax - sve.ymin;
            svw.size = svw.ymax - svw.ymin;
            // percept level location variables
            this.x = scne.x;
            this.y = scne.y;
            //percept level parts variable
            componentObjects.add(scne);
            componentObjects.add(scnw);
            componentObjects.add(scse);
            componentObjects.add(scsw);
            componentObjects.add(shn);
            componentObjects.add(shs);
            componentObjects.add(sve);
            componentObjects.add(svw);
        } // end constructor for two corners ne sw

....
//---------
    public void checkSquares() {
        LinkedList tentativeSquareList = new LinkedList();
        LinkedList corners = new LinkedList();
        for (int i=0; i<activeObjects.size(); i++) {
            Percept tempi = (Percept)activeObjects.get(i);
            if
((tempi.type=="cornerne")|(tempi.type=="cornerse")|(tempi.type=="cornersw")|(tempi.ty
pe=="cornernw")) {
                corners.add(tempi);
            } // end if corner
        } // end for i get corners
        // look for pairs of corners with proper alignment to be possible squares
        for (int aa=0; aa<corners.size(); aa++) {
            Percept tempa = (Percept)corners.get(aa);
            for (int bb=0; bb<corners.size(); bb++) {
                Percept tempb = (Percept)corners.get(bb);
                if
((tempa.type=="cornerne")&(tempb.type=="cornerse")&(tempa.x==tempb.x)&(tempa.y>
tempb.y)) {
                    CornerNE a = (CornerNE)tempa;
                    CornerSE b = (CornerSE)tempb;
                    Square squareab = new Square(a, b);
                    tentativeSquareList.add(squareab);
                } else if
((tempa.type=="cornerne")&(tempb.type=="cornernw")&(tempa.y==tempb.y)&(tempa.x
<tempb.x)){
                    CornerNE a = (CornerNE)tempa;
                    CornerNW b = (CornerNW)tempb;
                    Square squareab = new Square(a, b);
                    tentativeSquareList.add(squareab);
                } else if
((tempa.type=="cornerne")&(tempb.type=="cornersw")&(tempa.x<tempb.x)&(tempa.y>t
empb.y)) {
                    CornerNE a = (CornerNE)tempa;
                    CornerSW b = (CornerSW)tempb;
                    Square squareab = new Square(a, b);
                    tentativeSquareList.add(squareab);
                } else if
((tempa.type=="cornerse")&(tempb.type=="cornersw")&(tempa.y==tempb.y)&(tempa.x
<tempb.x)) {
                    CornerSE a = (CornerSE)tempa;
                    CornerSW b = (CornerSW)tempb;
                    Square squareab = new Square(a, b);
                    tentativeSquareList.add(squareab);
                } else if
((tempa.type=="cornerse")&(tempb.type=="cornernw")&(tempa.x<tempb.x)&(tempa.y<t
empb.y)) {
                    CornerSE a = (CornerSE)tempa;
                    CornerNW b = (CornerNW)tempb;
                    Square squareab = new Square(a, b);
                    tentativeSquareList.add(squareab);
                } else if
((tempa.type=="cornersw")&(tempb.type=="cornernw")&(tempa.x==tempb.x)&(tempa.y
<tempb.y)) {
                    CornerSW a = (CornerSW)tempa;
                    CornerNW b = (CornerNW)tempb;
                    Square squareab = new Square(a, b);
                    tentativeSquareList.add(squareab);
                }  // end if else pairing corners
                // no need to worry about crossing a corner with itself
            } // end for b
        } // end for a

        // remove redundant entries
        // use square.x, square.y, and square.offset to identify matches
        // note when one has been "removed" on an index
        // since removing the actual entry would shift the indices of the for loops
```

```
    // only consider entries which have not been "removed"
    int[] removed = new int[tentativeSquareList.size()];
    for (int k=0; k<tentativeSquareList.size(); k++) {
      removed[k] = 1;
    } // end for k intitialize index

    for (int i=0; i<tentativeSquareList.size(); i++) {
      Square tempi = (Square)tentativeSquareList.get(i);
      for (int j=0; j<tentativeSquareList.size(); j++) {
          Square tempj = (Square)tentativeSquareList.get(j);
          // don't remove if matching on the diagonal when crossing list by itself
          if (!(tempi==tempj)&(removed[i]==1)&(removed[j]==1)) {
            if
((tempi.x==tempj.x)&(tempi.y==tempj.y)&(tempi.offset==tempj.offset)) {
                //tentativeSquareList.remove(tempj);
                removed[j]=0;
            } // end if they match in location
            // also remove any "squares" with 0 offset
            if (tempi.offset==0) {
                //tentativeSquareList.remove(tempi);
                removed[i]=0;
            } // end if 0 offset
          } // end if they are not the same object
      } // end for j
    } // end for i

    // compile new list of good entries
    LinkedList goodSquares = new LinkedList();
    for (int r=0; r<tentativeSquareList.size(); r++) {
      Square tempr = (Square)tentativeSquareList.get(r);
      if (removed[r]==1) {
          goodSquares.add(tempr);
      } // end if
    } // end for r

    for (int k=0; k<goodSquares.size(); k++) {
      double match = 0;
      Square squarek = (Square)goodSquares.get(k);
      //System.out.println("squarek is " + squarek + squarek.x + squarek.y);
      //System.out.println("squarek parts list " + squarek.componentObjects);
      double partsn = squarek.componentObjects.size();
      for (int p=0; p<partsn; p++) {
          Percept tempp = (Percept)squarek.componentObjects.get(p);
          if (containsPercept(tempp, activeObjects)) {
            match++;
```

```
                //System.out.println("found part " + tempp.type + " at " + tempp.x + ","
+ tempp.y);
              } // end if match
          } // end for p parts
          double proportionMatched = match / partsn;
          // //System.out.println("proportion matched " + proportionMatched);
          if (proportionMatched>objectImplementationThreshold) {
              //System.out.println("matched " + proportionMatched + ", recognizing
square");
              //System.out.println(squarek.scne.x + "," + squarek.scne.y + " size: " +
squarek.offset);
              activateObject(squarek, proportionMatched*objectActivationWeight);
          } // end found enough parts
        } // end for k tentative squares
    } // end checkSquares
    //---------

    public void checkCubeA() {
        System.out.println("at check cube a");
        System.out.println("with active objects " + activeObjects);
        LinkedList tentativeCubeAList = new LinkedList();
        LinkedList squares = new LinkedList();
        LinkedList corners = new LinkedList();
        LinkedList hlines = new LinkedList();
        for (int i=0; i<activeObjects.size(); i++) {
          Percept tempi = (Percept)activeObjects.get(i);
          if (tempi.type=="square") {
              squares.add(tempi);
          } // end if square
          if (tempi.type=="cornerne3d") {
              corners.add(tempi);
          } // end if cne3d
          if (tempi.type=="lineh") {
              hlines.add(tempi);
          } // end if h
        } // end for i get squares, corners, horizontal lines

        // for cne, corner and square share x,y location
        for (int s =0; s<squares.size(); s++) {
          Square sq = (Square)squares.get(s);
          for (int c = 0; c<corners.size(); c++) {
              CornerNE3D c3 = (CornerNE3D)corners.get(c);
              if ((sq.x==c3.x)&(sq.y==c3.y)){
                for (int h = 0; h<hlines.size(); h++) {
                    HorizontalLine temph = (HorizontalLine)hlines.get(h);
                    //        hline y value must be more (below) square y value
```

```
                        if (temph.y>sq.y) {
                            CubeA newcube = new CubeA(sq, c3, temph);
                            tentativeCubeAList.add(newcube);
                            System.out.println("added possible cube " + newcube + " with
act " + newcube.act + " and off=" + newcube.off);
                        }
                    } // end for h
                } // end if square and corner are co-located
            } // end for c corners
        } // end for s squares

        // remove redundant entries
        // note when one has been "removed" on an index
        // since removing the actual entry would shift the indices of the for loops
        // only consider entries which have not been "removed"
        int[] removed = new int[tentativeCubeAList.size()];
        for (int k=0; k<tentativeCubeAList.size(); k++) {
            removed[k] = 1;
        } // end for k intitialize index

        for (int i=0; i<tentativeCubeAList.size(); i++) {
            CubeA tempi = (CubeA)tentativeCubeAList.get(i);
            for (int j=0; j<tentativeCubeAList.size(); j++) {
                    CubeA tempj = (CubeA)tentativeCubeAList.get(j);
                    // don't remove if matching on the diagonal when crossing list by itself
                    if (!(tempi==tempj)&(removed[i]==1)&(removed[j]==1)) {
                        if ((tempi.x==tempj.x)&(tempi.y==tempj.y)&(tempi.size==tempj.size))
{
                            //tentativeCubeAList.remove(tempj);
                            removed[j]=0;
                        } // end if they match in location
                    } // end if they are not the same object
            } // end for j
        } // end for i

        // compile new list of good entries
        LinkedList goodCubes = new LinkedList();
        for (int r=0; r<tentativeCubeAList.size(); r++) {
            CubeA tempr = (CubeA)tentativeCubeAList.get(r);
            if (removed[r]==1) {
                    goodCubes.add(tempr);
            } // end if
        } // end for r

        for (int k=0; k<goodCubes.size(); k++) {
            double match = 0;
            CubeA cubek = (CubeA)goodCubes.get(k);
            System.out.println("cubek is " + cubek + cubek.x + cubek.y);
            System.out.println("cubek parts list " + cubek.relatedObjects);
            double partsn = cubek.relatedObjects.size();

            for (int p=0; p<partsn; p++) {
                  Percept tempp = (Percept)cubek.relatedObjects.get(p);
                  if (containsPercept(tempp, activeObjects)) {
                     match++;
                     System.out.println("found part " + tempp.type + " at " + tempp.x + "," +
tempp.y);
                  } // end if match
            } // end for p parts
            //System.out.println("matched " + match + " parts");
            double proportionMatched = match / partsn;
            System.out.println("proportion matched is " + proportionMatched);
            if (proportionMatched>objectImplementationThreshold) {
                  System.out.println("matched " + proportionMatched + ", recognizing
CubeA");
                  activateConcept(cubek, proportionMatched*conceptActivationWeight);
            } // end found enough parts
        } // end for k tentative squares
    } // end checkCubeA

    //--------
    public void checkCubeB() {
        System.out.println("at check cube b");
        System.out.println("with active objects " + activeObjects);
        LinkedList tentativeCubeBList = new LinkedList();
        LinkedList squares = new LinkedList();
        LinkedList corners = new LinkedList();
        LinkedList hlines = new LinkedList();
        for (int i=0; i<activeObjects.size(); i++) {
           Percept tempi = (Percept)activeObjects.get(i);
           if (tempi.type=="square") {
                squares.add(tempi);
           } // end if square
           if (tempi.type=="cornersw3d") {
                corners.add(tempi);
           } // end if csw3d
           if (tempi.type=="lineh") {
                hlines.add(tempi);
           } // end if h
        } // end for i get squares, corners, horizontal lines

        // for csw, corner xy is offset from square xy by size of square
```

```
for (int s =0; s<squares.size(); s++) {
   Square sq = (Square)squares.get(s);
   for (int c = 0; c<corners.size(); c++) {
         CornerSW3D c3 = (CornerSW3D)corners.get(c);
         if ((sq.x==c3.x - sq.size)&(sq.y==c3.y + sq.size)){
            for (int h = 0; h<hlines.size(); h++) {
                  HorizontalLine temph = (HorizontalLine)hlines.get(h);
                  //        hline y value must be less (above) square y value
                  if (temph.y<sq.y) {
                     //System.out.println("possible cube");
                     CubeB newcube = new CubeB(sq, c3, temph);
                     tentativeCubeBList.add(newcube);
                  }
            } // end for h
         } // end if square and corner are co-located
   } // end for c corners
} // end for s squares

// remove redundant entries
// note when one has been "removed" on an index
// since removing the actual entry would shift the indices of the for loops
// only consider entries which have not been "removed"
int[] removed = new int[tentativeCubeBList.size()];
for (int k=0; k<tentativeCubeBList.size(); k++) {
   removed[k] = 1;
} // end for k intitialize index

for (int i=0; i<tentativeCubeBList.size(); i++) {
   CubeB tempi = (CubeB)tentativeCubeBList.get(i);
   for (int j=0; j<tentativeCubeBList.size(); j++) {
         CubeB tempj = (CubeB)tentativeCubeBList.get(j);
         // don't remove if matching on the diagonal when crossing list by itself
         if (!(tempi==tempj)&(removed[i]==1)&(removed[j]==1)) {
            if ((tempi.x==tempj.x)&(tempi.y==tempj.y)&(tempi.size==tempj.size))
{
                  removed[j]=0;
            } // end if they match in location
         } // end if they are not the same object
   } // end for j
} // end for i

// compile new list of good entries
LinkedList goodCubes = new LinkedList();
for (int r=0; r<tentativeCubeBList.size(); r++) {
   CubeB tempr = (CubeB)tentativeCubeBList.get(r);
   if (removed[r]==1) {
```

```
                goodCubes.add(tempr);
            } // end if
        } // end for r

        for (int k=0; k<goodCubes.size(); k++) {
            double match = 0;
            CubeB cubek = (CubeB)goodCubes.get(k);
            System.out.println("cubek is " + cubek + cubek.x + cubek.y);
            System.out.println("cubek parts list " + cubek.relatedObjects);
            double partsn = cubek.relatedObjects.size();

            for (int p=0; p<partsn; p++) {
                    Percept tempp = (Percept)cubek.relatedObjects.get(p);
                    if (containsPercept(tempp, activeObjects)) {
                        match++;
                        System.out.println("found part " + tempp.type + " at " + tempp.x + "," +
tempp.y);
                    } // end if match
            } // end for p parts
            System.out.println("matched " + match + " parts");
            double proportionMatched = match / partsn;
            //System.out.println("proportion matched is " + proportionMatched);
            if (proportionMatched>objectImplementationThreshold) {
                    System.out.println("matched " + proportionMatched + ", recognizing
CubeB");
                    activateConcept(cubek, proportionMatched*conceptActivationWeight);
            } // end found enough parts
        } // end for k tentative squares
    } // end checkCubeB
    // ----
public LinkedList deriveSquare3(FeatureDetector f) {
        LinkedList squares = new LinkedList();
        FeatureDetector output = new FeatureDetector();
        Square3 temp = new Square3();
        int[] a = f.inputCode;
        if (a==cne) {
            output.x = f.x;
            output.y = f.y;
            temp = new Square3(output);
            squares.add(temp);
        } else if (a==cse) {
            output.x = f.x;
            output.y = f.y + 6;
            temp = new Square3(output);
            squares.add(temp);
        } else if (a==csw) {
```

```
      output.x = f.x - 6;
      output.y = f.y + 6;
      temp = new Square3(output);
      squares.add(temp);
    } else if (a==cnw) {
      output.x = f.x - 6;
      output.y = f.y;
      temp = new Square3(output);
      squares.add(temp);
    } // end if else
    return squares;
} // end deriveSquare3

//------------
public void checkPartsSquare3(LinkedList inputFeatures) {
    // identify matching parts
    for (int f=0; f<inputFeatures.size(); f++) {
      FeatureDetector tempf = (FeatureDetector)inputFeatures.get(f);
    } // end f print
    LinkedList matchingParts = new LinkedList();
    LinkedList targetParts = new LinkedList();
    targetParts.add(csw);
    targetParts.add(cnw);
    targetParts.add(cne);
    targetParts.add(cse);
    targetParts.add(v);
    targetParts.add(v);
    targetParts.add(h);
    targetParts.add(h);
    for (int i=0; i<targetParts.size(); i++) {
      int[] targeti = (int[])targetParts.get(i);
      // int[] targetBits = targeti.inputCode;
      for (int j=0; j<inputFeatures.size(); j++) {
          FeatureDetector inputj = (FeatureDetector)inputFeatures.get(j);
          int[] inputBits = inputj.inputCode;
          if (inputBits==targeti) {
            matchingParts.add(inputj);
          } // end if input contains target part
      } // end for j input features
    } // end for i target features

    // if enough matching parts, take each matching part in turn
    LinkedList allSquares = new LinkedList();
    if (matchingParts.size()>postulationThreshold) {
      for(int k=0; k<matchingParts.size(); k++) {
          FeatureDetector matchingk = (FeatureDetector)matchingParts.get(k);
```

```
                //call function to create template from this feature and location
                LinkedList squaresk = deriveSquare3(matchingk);
                allSquares.addAll(squaresk);
            } // end for k matching parts
            // remove redundant entries
            // use square.x and square.y to identify matches
            for (int i=0; i<allSquares.size(); i++) {
                Square3 tempi = (Square3)allSquares.get(i);
                for (int j=0; j<allSquares.size(); j++) {
                   Square3 tempj = (Square3)allSquares.get(j);
                   // don't remove if matching on the diagonal when crossing list by itself
                   if (!(tempi==tempj)) {
                        if ((tempi.x==tempj.x) & (tempi.y==tempj.y)) {
                           allSquares.remove(tempj);
                        } // end if they match in location
                   } // end if they are not the same object
                } // end for j
            } // end for i

            // for each square on list of potential squares from single feature input,
            for (int n=0; n<allSquares.size(); n++) {
                Square3 squaren = (Square3)allSquares.get(n);
                //check input against specified features
                //use inputFeatures list
                LinkedList partsn = squaren.features;
                for (int k=0; k<partsn.size(); k++) {
                   FeatureDetector partk = (FeatureDetector)partsn.get(k);
                } // end for k print
                double match = 0;
                for (int p=0; p<partsn.size(); p++) {
                   FeatureDetector tempp = (FeatureDetector)partsn.get(p);
                   if (containsff(tempp, inputFeatures)) {
                        match++;
                   }
                } // end for p parts in template
                double proportionMatched = match / partsn.size();
                //if enough input matches, create object and assign activation accordingly
                //also need to put object on recognition list, masterObjectList
                if (proportionMatched > objectImplementationThreshold) {
                   activateObject(squaren, proportionMatched*objectActivationWeight);
                   square3List.add(squaren);
                } // end if matching x/8 parts
            } // end for n allSquares
        } // end if enough matching parts for percept
    } // end checkPartsSquare3
```

```
  //-------
  // consolidation methods combine adjacent line objects
  // interrupted lines may be consolidated because of feature-level tolerance of missing
bits,
  // and because of object level methods that extend the lines by one bit

  public LinkedList consolidateDiagonalSWNELines(LinkedList activeF) {
        LinkedList objects1 =           consDiagonalSWNEFeatures(activeF);
        LinkedList objects2 = consDiagonalSWNELineObjects(objects1);
        LinkedList objectsFinal = removeRedundantDiagonalSWNELines(objects2);
        for (int i=0; i<objectsFinal.size(); i++) {
           Percept percepti = (Percept)objectsFinal.get(i);
           activateObject(percepti, objectActivationWeight);
        } // end for i give activation to recognized objects
        return objectsFinal;
  } //end consoldicateDiagonalSWNELines

  public LinkedList consDiagonalSWNEFeatures(LinkedList input) {
        LinkedList output = new LinkedList();
        // find diagonal line features; screen nulls
        LinkedList dlines = new LinkedList();
        for (int a=0; a<input.size(); a++) {
           FeatureDetector tempa = (FeatureDetector)input.get(a);
           if ((tempa!=null)&(tempa.inputCode==diagswne)){
                 dlines.add(tempa);
           } // end if tempa is diagonal line
        } // end for a copy list

        // assume segments are orphans (1) unless proved otherwise
        // note that this status does NOT affect whether segment can participate in other
pairings
        int[] status = new int[dlines.size()];
        for (int b=0; b<dlines.size(); b++) {
           status[b] = 1;
        } // end for b initialize status index

        for (int i=0; i<dlines.size(); i++) {
           FeatureDetector tempi = (FeatureDetector)dlines.get(i);
           for (int j=0; j<dlines.size(); j++) {
                 // this clause ensures that we don't compare an entry to itself (i=j)
                 // and that we don't check pairs ji that have already been checked as ij
                 if(j>i) {
                    FeatureDetector tempj = (FeatureDetector)dlines.get(j);
                    //lines must have same x and y displacement to be part of same line
                    // also, centers must be within range
                    int xdisp = Math.abs(tempi.x - tempj.x);
                    int ydisp = Math.abs(tempi.y - tempj.y);
                    if (xdisp==ydisp){
                        // centers must be within 3 bits on x-axis (i.e, edges touching)
                        if (xdisp<4) {
                            // ordering of segments on x-axis is handled by constructor
                            DiagonalLineSWNE newLine = new
DiagonalLineSWNE(tempi, tempj);
                            output.add(newLine);
                            diagonalSWNELineObjectList.add(newLine);
                            //record that these line segments have been used at least once to
create larger line objects
                            status[i]=0;
                            status[j]=0;
                        } // end if close enough
                    } // end compare
                } // end if j > i
            } // end for j
        } // end for i

        for (int k=0; k<dlines.size(); k++) {
            FeatureDetector tempk = (FeatureDetector)dlines.get(k);
            if (status[k]==1) {
                DiagonalLineSWNE orphanLine = new DiagonalLineSWNE(tempk);
                output.add(orphanLine);
            } // end if single
        } // end for k get singles
        return output;
    } // end consDiagonalSWNEFeatures
    //--------

    // if line objects are displaced by an equal amount for x and y,
    // and edges touch, combine them into one larger digaonal line object
    // use status index to keep track of any that are not used, and add them to list as singles
    public LinkedList consDiagonalSWNELineObjects (LinkedList input) {
        boolean goon = true;
        LinkedList output = new LinkedList();
        LinkedList copy = new LinkedList();
        for (int a=0; a<input.size(); a++) {
            DiagonalLineSWNE tempa = (DiagonalLineSWNE)input.get(a);
            copy.add(tempa);
        } // end for a copy list

        // 1 indicates need to include it as a single
        int[] status = new int[copy.size()];
        for (int b=0; b<copy.size(); b++) {
            status[b] = 1;
        } // end for b initialize status index

        for (int i=0; i<copy.size(); i++) {
           DiagonalLineSWNE tempi = (DiagonalLineSWNE)copy.get(i);
           for (int j=0; j<copy.size(); j++) {
                 // this clause ensures that we don't compare an entry to itself (i=j)
                 // and that we don't check pairs ji that have already been checked as ij
                 if(j>i){
                    DiagonalLineSWNE tempj = (DiagonalLineSWNE)copy.get(j);
                    // lines must have same x and y displacement
                    //ignore lines with a common endpoint,a.xmin=b.xmin, or
a.xmax=b.xmax :  larger line covers it
                    //ignore lines subsumed within the other line, a.xmin<b.xmin AND
a.xmax>b.xmax :  a covers it
                    int xdisp = Math.abs(tempi.x - tempj.x);
                    int ydisp = Math.abs(tempi.y - tempj.y);
                    if (xdisp==ydisp) {
                          goon = true;
                          if ((tempi.xmin==tempj.xmin)&(tempi.ymax==tempj.ymax)) {
                             goon = false;
                          } // end if lines share sw endpoint (sw = xmin, ymax)
                          if ((tempi.xmax==tempj.xmax)&(tempi.ymin==tempj.ymin)) {
                             goon = false;
                          } // end if lines share ne endpoint (ne = xmax, ymin)

                          // sufficient to check containment in x coordinates
                          if ((tempi.xmin<tempj.xmin)&(tempi.xmax>tempj.xmax)) {
                             goon = false;
                             //first line is bigger on both ends
                          } // end if a contains b
                          if ((tempj.xmin<tempi.xmin)&(tempj.xmax>tempi.xmax)) {
                             //second line is bigger on both ends
                             goon = false;
                          } // end if b contains a

                          // calculate gap from left-max to right-min
                          // adjacent lines will have distance = 1 (e.g., left-max = 3, right-
min = 4)

                          if (goon) {
                             int distance = 100;
                             if (tempi.xmin<tempj.xmin) {
                                   distance = tempj.xmin - tempi.xmax;
                             } // end if i..j
                             if (tempj.xmin<tempi.xmin) {
                                   distance = tempi.xmin - tempj.xmax;
                             } // end if j..i
```

```
                        // combine lines, set status for each part to 0
                        if (distance<2) {
                            DiagonalLineSWNE newLine = new
DiagonalLineSWNE(tempi, tempj);
                            diagonalSWNELineObjectList.add(newLine);
                            output.add(newLine);
                            status[i]=0;
                            status[j]=0;
                        } // end if adjacent
                    } // end if goon
                }// end if same y location
            } // end if j > i
        } // end for j
    } // end for i

    for (int k=0; k<copy.size(); k++) {
        DiagonalLineSWNE tempk = (DiagonalLineSWNE)copy.get(k);
        if (status[k]==1) {
            output.add(tempk);
        } // end if single
    } // end for k get singles
    return output;
} // end consDiagonalSWNELineObjects

public LinkedList removeRedundantDiagonalSWNELines (LinkedList input) {
    LinkedList output = new LinkedList();
    LinkedList copy = new LinkedList();
    for (int a=0; a<input.size(); a++) {
        DiagonalLineSWNE tempa = (DiagonalLineSWNE)input.get(a);
        copy.add(tempa);
    } // end for a copy list
    // 0 indicates a duplicate
    int[] status = new int[copy.size()];
    for (int b=0; b<copy.size(); b++) {
        status[b] = 1;
    } // end for b initialize status index
    for (int i=0; i<copy.size(); i++) {
        DiagonalLineSWNE tempi = (DiagonalLineSWNE)copy.get(i);
        for (int j=0; j<copy.size(); j++) {
            // this clause ensures that we don't compare an entry to itself (i=j)
            // and that we don't check pairs ji that have already been checked as ij
            if(j>i) {
                DiagonalLineSWNE tempj = (DiagonalLineSWNE)copy.get(j);
                // must have same x, y displacement
                // sw corner has xmin, ymax values
```

```
                    int xdisp = Math.abs(tempi.xmin - tempj.xmin);
                    int ydisp = Math.abs(tempi.ymax - tempj.ymax);
                    if (xdisp==ydisp) {
                        // if same location, arbitrarily remove second entry
                        // if tempi contains tempj, remove tempj
                        // if tempj contains tempi, remove tempi
                        if ((tempi.xmin==tempj.xmin)&(tempi.xmax==tempj.xmax)){
                          status[j]=0;
                        } else if
((tempi.xmin<=tempj.xmin)&(tempi.xmax>=tempj.xmax)) {
                          status[j]=0;
                        } else if
((tempj.xmin<=tempi.xmin)&(tempj.xmax>=tempi.xmax)) {
                          status[i]=0;
                        } // end if else else
                    } // end if y = y
                  } // end if j > i
              } // end for j
            } // end for i

            // put in output list any lines which have not been declared duplicates
            for (int k=0; k<copy.size(); k++) {
              DiagonalLineSWNE tempk = (DiagonalLineSWNE)copy.get(k);
              if (status[k]==1) {
                  output.add(tempk);
              } // end if single
            } // end for k get singles

            return output;
        } // end removeRedundantDiagonalSWNELines

        //-------
        public LinkedList consolidateDiagonalNWSELines(LinkedList activeF) {
            LinkedList objects1 =          consDiagonalNWSEFeatures(activeF);
            LinkedList objects2 = consDiagonalNWSELineObjects(objects1);
            LinkedList objectsFinal = removeRedundantDiagonalNWSELines(objects2);
            for (int i=0; i<objectsFinal.size(); i++) {
              Percept percepti = (Percept)objectsFinal.get(i);
              activateObject(percepti, objectActivationWeight);
            } // end for i give activation to recognized objects
            return objectsFinal;
        }  //end consoldicateDiagonalNWSELines

        public LinkedList consDiagonalNWSEFeatures(LinkedList input) {
            LinkedList output = new LinkedList();
            // find diagonal line features; screen nulls
        LinkedList dlines = new LinkedList();
        for (int a=0; a<input.size(); a++) {
           FeatureDetector tempa = (FeatureDetector)input.get(a);
           if ((tempa!=null)&(tempa.inputCode==diagnwse)){
                 dlines.add(tempa);
           } // end if tempa is diagonal line
        } // end for a copy list

        // assume segments are orphans (1) unless proved otherwise
        // note that this status does NOT affect whether segment can participate in other
pairings
        int[] status = new int[dlines.size()];
        for (int b=0; b<dlines.size(); b++) {
           status[b] = 1;
        } // end for b initialize status index

        for (int i=0; i<dlines.size(); i++) {
           FeatureDetector tempi = (FeatureDetector)dlines.get(i);
           for (int j=0; j<dlines.size(); j++) {
                 // this clause ensures that we don't compare an entry to itself (i=j)
                 // and that we don't check pairs ji that have already been checked as ij
                 if(j>i) {
                    FeatureDetector tempj = (FeatureDetector)dlines.get(j);
                    //lines must have same x and y displacement to be part of same line
                    // also, centers must be within range
                    int xdisp = Math.abs(tempi.x - tempj.x);
                    int ydisp = Math.abs(tempi.y - tempj.y);
                    if (xdisp==ydisp){
                          // centers must be within 3 bits on x-axis (i.e, edges touching)
                          if (xdisp<4) {
                             // ordering of segments on x-axis is handled by constructor
                             DiagonalLineNWSE newLine = new
DiagonalLineNWSE(tempi, tempj);
                             output.add(newLine);
                             diagonalNWSELineObjectList.add(newLine);
                             //record that these line segments have been used at least once to
create larger line objects
                             status[i]=0;
                             status[j]=0;
                          } // end if close enough
                    } // end compare
                 } // end if j > i
           } // end for j
        } // end for i

        for (int k=0; k<dlines.size(); k++) {
            FeatureDetector tempk = (FeatureDetector)dlines.get(k);
            if (status[k]==1) {
                  DiagonalLineNWSE orphanLine = new DiagonalLineNWSE(tempk);
                  output.add(orphanLine);
            } // end if single
        } // end for k get singles
        return output;
    } // end consDiagonalNWSEFeatures

    // if line objects are displaced by an equal amount for x and y,
    // and edges touch, combine them into one larger diagonal line object
    // use status index to keep track of any that are not used, and add them to list as singles
    public LinkedList consDiagonalNWSELineObjects (LinkedList input) {
        boolean goon = true;
        LinkedList output = new LinkedList();
        LinkedList copy = new LinkedList();
        for (int a=0; a<input.size(); a++) {
            DiagonalLineNWSE tempa = (DiagonalLineNWSE)input.get(a);
            copy.add(tempa);
        } // end for a copy list

        // 1 indicates need to include it as a single
        int[] status = new int[copy.size()];
        for (int b=0; b<copy.size(); b++) {
            status[b] = 1;
        } // end for b initialize status index

        for (int i=0; i<copy.size(); i++) {
            DiagonalLineNWSE tempi = (DiagonalLineNWSE)copy.get(i);
            for (int j=0; j<copy.size(); j++) {
                  // this clause ensures that we don't compare an entry to itself (i=j)
                  // and that we don't check pairs ji that have already been checked as ij
                  if(j>i){
                     DiagonalLineNWSE tempj = (DiagonalLineNWSE)copy.get(j);
                     // lines must have same x and y displacement
                     //ignore lines with a common endpoint,a.xmin=b.xmin, or
a.xmax=b.xmax :  larger line covers it
                     //ignore lines subsumed within the other line, a.xmin<b.xmin AND
a.xmax>b.xmax :  a covers it
                     int xdisp = Math.abs(tempi.x - tempj.x);
                     int ydisp = Math.abs(tempi.y - tempj.y);
                     if (xdisp==ydisp) {
                           goon = true;
                           if ((tempi.xmin==tempj.xmin)&(tempi.ymin==tempj.ymin)) {
                              goon = false;
                           } // end if lines share nw endpoint (nw = xmin, ymin)
```

```
                    if ((tempi.xmax==tempj.xmax)&(tempi.ymax==tempj.ymax)) {
                      goon = false;
                    } // end if lines share se endpoint (se = xmax, ymax)
                    // sufficient to check containment in x coordinates
                    if ((tempi.xmin<tempj.xmin)&(tempi.xmax>tempj.xmax)) {
                      goon = false;
                    } // end if a contains b
                    if ((tempj.xmin<tempi.xmin)&(tempj.xmax>tempi.xmax)) {
                      goon = false;
                    } // end if b contains a

                    // calculate gap from left-max to right-min
                    // adjacent lines will have distance = 1 (e.g., left-max = 3, right-
min = 4)
                    if (goon) {
                      int distance = 100;
                      if (tempi.xmin<tempj.xmin) {
                          distance = tempj.xmin - tempi.xmax;
                      } // end if i..j
                      if (tempj.xmin<tempi.xmin) {
                          distance = tempi.xmin - tempj.xmax;
                      } // end if j..i

                      // combine lines, set status for each part to 0
                      if (distance<2) {
                          DiagonalLineNWSE newLine = new
DiagonalLineNWSE(tempi, tempj);
                          diagonalNWSELineObjectList.add(newLine);
                          output.add(newLine);
                          status[i]=0;
                          status[j]=0;
                      } // end if adjacent
                    } // end if goon
                }// end if same y location
              } // end if j > i
          } // end for j
        } // end for i

        for (int k=0; k<copy.size(); k++) {
          DiagonalLineNWSE tempk = (DiagonalLineNWSE)copy.get(k);
          if (status[k]==1) {
              output.add(tempk);
          } // end if single
        } // end for k get singles
        return output;
    } // end consDiagonalNWSELineObjects
```

```
   public LinkedList removeRedundantDiagonalNWSELines (LinkedList input) {
      LinkedList output = new LinkedList();
      LinkedList copy = new LinkedList();
      for (int a=0; a<input.size(); a++) {
         DiagonalLineNWSE tempa = (DiagonalLineNWSE)input.get(a);
         copy.add(tempa);
      } // end for a copy list
      // 0 indicates a duplicate
      int[] status = new int[copy.size()];
      for (int b=0; b<copy.size(); b++) {
         status[b] = 1;
      } // end for b initialize status index
      for (int i=0; i<copy.size(); i++) {
         DiagonalLineNWSE tempi = (DiagonalLineNWSE)copy.get(i);
         for (int j=0; j<copy.size(); j++) {
               // this clause ensures that we don't compare an entry to itself (i=j)
               // and that we don't check pairs ji that have already been checked as ij
               if(j>i) {
                  DiagonalLineNWSE tempj = (DiagonalLineNWSE)copy.get(j);
                  // must have same x, y displacement
                  // nw corner has xmin, ymin values
                  int xdisp = Math.abs(tempi.xmin - tempj.xmin);
                  int ydisp = Math.abs(tempi.ymin - tempj.ymin);
                  if (xdisp==ydisp) {
                        // if same location, arbitrarily remove second entry
                        // if tempi contains tempj, remove tempj
                        // if tempj contains tempi, remove tempi
                        if ((tempi.xmin==tempj.xmin)&(tempi.xmax==tempj.xmax)){
                           status[j]=0;
                        } else if
((tempi.xmin<=tempj.xmin)&(tempi.xmax>=tempj.xmax)) {
                           status[j]=0;
                        } else if
((tempj.xmin<=tempi.xmin)&(tempj.xmax>=tempi.xmax)) {
                           status[i]=0;
                        } // end if else else
                  } // end if y = y
               } // end if j > i
         } // end for j
      } // end for i

      // put in output list any lines which have not been declared duplicates
      for (int k=0; k<copy.size(); k++) {
         DiagonalLineNWSE tempk = (DiagonalLineNWSE)copy.get(k);
         if (status[k]==1) {
                output.add(tempk);
            } // end if single
        } // end for k get singles
        return output;
    } // end removeRedundantDiagonalNWSELines
    //---------

    // method to extend vertical line in either direction,
    // provided bit map does not contradict extension (overlay OK)
    public LinkedList extendVerticalLines(LinkedList lines) {
        LinkedList extendedLines = new LinkedList();
        for (int i=0; i<lines.size(); i++) {
            VerticalLine v = (VerticalLine)lines.get(i);
            int ymax = v.ymax + 1;
            int ymin = v.ymin -1;
            int x = v.x;
            // consider feature x, ymax, v  (code 0=v)
            if (ymax<screenymax) {
                FeatureDetector fdvymax = featureListMatrix[x][ymax][0];
                BitGrid bymax = bitGridMatrix[x][ymax];
                double maxmatch = compareWithOverlay(fdvymax.inputCode,
bymax.input);
                if (maxmatch > overlayThreshold) {
                    v.ymax = ymax;
                    // this extends the length of the horizontal line object
                } // end if over threshold
            } // end if within screenymax
            // consider feature xmin, y, h
            if (ymin>screenymin) {
                FeatureDetector fdvymin = featureListMatrix[x][ymin][0];
                BitGrid bymin = bitGridMatrix[x][ymin];
                double minmatch = compareWithOverlay(fdvymin.inputCode,
bymin.input);
                if (minmatch > overlayThreshold) {
                    v.ymin = ymin;
                    // this extends the length of the horizontal line object
                } // end if over threshold
            } // end if within screenymin
        } // end for i input lines
        return lines;
    } // end extendVerticalLines
    //------------

    // method to extend horizontal line in either direction,
    // provided bit map does not contradict extension (overlay OK)
    public LinkedList extendHorizontalLines(LinkedList lines) {
        LinkedList extendedLines = new LinkedList();
        for (int i=0; i<lines.size(); i++) {
          HorizontalLine h = (HorizontalLine)lines.get(i);
          int xmax = h.xmax + 1;
          int xmin = h.xmin -1;
          int y = h.y;
          // consider feature xmax, y, h  (code 1=h)
          if (xmax<screenxmax) {
               FeatureDetector fdhxmax = featureListMatrix[xmax][y][1];
               BitGrid bxmax = bitGridMatrix[xmax][y];
               double maxmatch = compareWithOverlay(fdhxmax.inputCode,
bxmax.input);
               if (maxmatch > overlayThreshold) {
                 h.xmax = xmax;
                 // this extends the length of the horizontal line object
               } // end if over threshold
          } // end if within screenxmax
          // consider feature xmin, y, h

          if (xmin>screenxmin) {
               FeatureDetector fdhxmin = featureListMatrix[xmin][y][1];
               BitGrid bxmin = bitGridMatrix[xmin][y];
               double minmatch = compareWithOverlay(fdhxmin.inputCode,
bxmin.input);
               if (minmatch > overlayThreshold) {
                 h.xmin = xmin;
                 // this extends the length of the horizontal line object
               } // end if over threshold
          } // end if within screenxmin
        } // end for i input lines
        return lines;
    } // end extendHorizontalLines

    //------------
    public LinkedList consolidateHorizontalLines(LinkedList activeF) {
        LinkedList objects1 =           consHorizontalFeatures(activeF);
        LinkedList objects2 = consHorizontalLineObjects(objects1);
        LinkedList objectsFinal = removeRedundantHorizontalLines(objects2);
        LinkedList extendedHorizontalLines = extendHorizontalLines(objectsFinal);
        LinkedList consExtended =
consHorizontalLineObjects(extendedHorizontalLines);
        LinkedList nonRedundantExtended =
removeRedundantHorizontalLines(consExtended);
        // now activate lines discovered
        for (int i=0; i<nonRedundantExtended.size(); i++) {
          Percept percepti = (Percept)nonRedundantExtended.get(i);
```

```
            activateObject(percepti, objectActivationWeight);
        } // end for i give activation to recognized objects
        return nonRedundantExtended;
    } //end consoldicateHorizontalLines

    public LinkedList consHorizontalFeatures(LinkedList input) {
        LinkedList output = new LinkedList();
        // find horizontal line features; screen nulls
        LinkedList hlines = new LinkedList();
        for (int a=0; a<input.size(); a++) {
            FeatureDetector tempa = (FeatureDetector)input.get(a);
            if ((tempa!=null)&(tempa.inputCode==h)){
                hlines.add(tempa);
            } // end if tempa is horizontal line
        } // end for a copy list

        // assume segments are orphans (1) unless proved otherwise
        // note that this status does NOT affect whether segment can participate in other
pairings
        int[] status = new int[hlines.size()];
        for (int b=0; b<hlines.size(); b++) {
            status[b] = 1;
        } // end for b initialize status index

        for (int i=0; i<hlines.size(); i++) {
            FeatureDetector tempi = (FeatureDetector)hlines.get(i);
            for (int j=0; j<hlines.size(); j++) {
                // this clause ensures that we don't compare an entry to itself (i=j)
                // and that we don't check pairs ji that have already been checked as ij
                if(j>i) {
                    FeatureDetector tempj = (FeatureDetector)hlines.get(j);
                    // must be at same y coordinates, but distinct x coordinates
                    if ((tempi.y==tempj.y)&(tempi.x!=tempj.x)){
                        // centers must be within 3 bits on x-axis (i.e, edges touching)
                        int centerDistance = Math.abs(tempi.x - tempj.x);
                        if (centerDistance<4) {
                            // ordering of segments on x-axis is handled by constructor
                            HorizontalLine newLine = new HorizontalLine(tempi, tempj);
                            output.add(newLine);
                            horizontalLineObjectList.add(newLine);
                            //record that these line segments have been used at least once to
create larger line objects
                            status[i]=0;
                            status[j]=0;
                        } // end if close enough
                    } // end compare
            } // end if j > i
        } // end for j
    } // end for i

    for (int k=0; k<hlines.size(); k++) {
        FeatureDetector tempk = (FeatureDetector)hlines.get(k);
        if (status[k]==1) {
            HorizontalLine orphanLine = new HorizontalLine(tempk);
            output.add(orphanLine);
            horizontalLineObjectList.add(orphanLine);
        } // end if single
    } // end for k get singles
    return output;
} // end consHorizontalFeatures

//-------
// if line objects are in same y location, and edges touch, combine them into one larger
line object
// use status index to keep track of any that are not used, and add them to list as singles
public LinkedList consHorizontalLineObjects (LinkedList input) {
    boolean goon = true;
    LinkedList output = new LinkedList();
    LinkedList copy = new LinkedList();
    for (int a=0; a<input.size(); a++) {
        HorizontalLine tempa = (HorizontalLine)input.get(a);
        copy.add(tempa);
    } // end for a copy list

    // 1 indicates need to include it as a single
    int[] status = new int[copy.size()];
    for (int b=0; b<copy.size(); b++) {
        status[b] = 1;
    } // end for b initialize status index

    for (int i=0; i<copy.size(); i++) {
        HorizontalLine tempi = (HorizontalLine)copy.get(i);
        for (int j=0; j<copy.size(); j++) {
            // this clause ensures that we don't compare an entry to itself (i=j)
            // and that we don't check pairs ji that have already been checked as ij
            if(j>i){
                HorizontalLine tempj = (HorizontalLine)copy.get(j);
                // lines must share same y-coordinate
                //ignore lines with a common endpoint,a.xmin=b.xmin, or
a.xmax=b.xmax :  larger line covers it
                //ignore lines subsumed within the other line, a.xmin<b.xmin AND
a.xmax>b.xmax :  a covers it
```

```
                    if (tempi.y==tempj.y) {
                        goon = true;
                        if ((tempi.xmin==tempj.xmin)|(tempi.xmax==tempj.xmax)) {
                           goon = false;
                        } // end if lines share an endpoint
                        if ((tempi.xmin<tempj.xmin)&(tempi.xmax>tempj.xmax)) {
                           goon = false;
                        } // end if a contains b
                        if ((tempj.xmin<tempi.xmin)&(tempj.xmax>tempi.xmax)) {
                           goon = false;
                        } // end if b contains a

                        // calculate gap from left-max to right-min
                        // adjacent lines will have distance = 1 (e.g., left-max = 3, right-
min = 4)
                        if (goon) {
                           int distance = 100;
                           if (tempi.xmin<tempj.xmin) {
                                 distance = tempj.xmin - tempi.xmax;
                           } // end if i..j
                           if (tempj.xmin<tempi.xmin) {
                                 distance = tempi.xmin - tempj.xmax;
                           } // end if j..i
                           // combine lines, set status for each part to 0
                           if (distance<2) {
                                 HorizontalLine newLine = new HorizontalLine(tempi,
tempj);

                                 horizontalLineObjectList.add(newLine);
                                 output.add(newLine);
                                 status[i]=0;
                                 status[j]=0;
                           } // end if adjacent
                        } // end if goon
                    }// end if same y location
                 } // end if j > i
            } // end for j
         } // end for i

         for (int k=0; k<copy.size(); k++) {
            HorizontalLine tempk = (HorizontalLine)copy.get(k);
            if (status[k]==1) {
                  output.add(tempk);
            } // end if single
         } // end for k get singles

         return output;
```

```
} // end consHorizontalLineObjects
//-------

public LinkedList removeRedundantHorizontalLines (LinkedList input) {
      LinkedList output = new LinkedList();
      LinkedList copy = new LinkedList();
      for (int a=0; a<input.size(); a++) {
         HorizontalLine tempa = (HorizontalLine)input.get(a);
         copy.add(tempa);
      } // end for a copy list
      // 0 indicates a duplicate
      int[] status = new int[copy.size()];
      for (int b=0; b<copy.size(); b++) {
         status[b] = 1;
      } // end for b initialize status index
      for (int i=0; i<copy.size(); i++) {
         HorizontalLine tempi = (HorizontalLine)copy.get(i);
         for (int j=0; j<copy.size(); j++) {
               // this clause ensures that we don't compare an entry to itself (i=j)
               // and that we don't check pairs ji that have already been checked as ij
               if(j>i) {
                  HorizontalLine tempj = (HorizontalLine)copy.get(j);
                  // must be in same y location
                  if (tempi.y==tempj.y) {
                        // if same location, arbitrarily remove second entry
                        // if tempi contains tempj, remove tempj
                        // if tempj contains tempi, remove tempi
                        if
((tempi.y==tempj.y)&(tempi.xmin==tempj.xmin)&(tempi.xmax==tempj.xmax)){
                              status[j]=0;
                        } else if
((tempi.xmin<=tempj.xmin)&(tempi.xmax>=tempj.xmax)) {
                              status[j]=0;
                        } else if
((tempj.xmin<=tempi.xmin)&(tempj.xmax>=tempi.xmax)) {
                              status[i]=0;
                        } // end if else else
                  } // end if y = y
               } // end if j > i
         } // end for j
      } // end for i

      // put in output list any lines which have not been declared duplicates
      for (int k=0; k<copy.size(); k++) {
         HorizontalLine tempk = (HorizontalLine)copy.get(k);
         if (status[k]==1) {
                output.add(tempk);
            } // end if single
        } // end for k get singles
        return output;
    } // end removeRedundantHorizontalLines
    //----------------------

    // once consolidated, call activateObject on each unique line
    // activateObjects either updates an existing object by incrementing activation,
    // or creates a new object with initial activation
    public LinkedList consolidateVerticalLines(LinkedList activeF) {
        LinkedList objects1 =          consVerticalFeatures(activeF);
        LinkedList objects2 = consVerticalLineObjects(objects1);
        LinkedList objectsFinal = removeRedundantVerticalLines(objects2);
        LinkedList extendedVerticalLines = extendVerticalLines(objectsFinal);
        LinkedList consExtended = consVerticalLineObjects(extendedVerticalLines);
        LinkedList nonRedundantExtended =
removeRedundantVerticalLines(consExtended);
        // now activate discovered lines
        for (int i=0; i<nonRedundantExtended.size(); i++) {
            Percept percepti = (Percept)nonRedundantExtended.get(i);
            activateObject(percepti, objectActivationWeight);
        } // end for i give activation to recognized objects
        return nonRedundantExtended;
    } //end consoldicateVerticalLines

    public LinkedList consVerticalFeatures(LinkedList input) {
        LinkedList output = new LinkedList();
        // find vertical line features; screen nulls
        LinkedList vlines = new LinkedList();
        for (int a=0; a<input.size(); a++) {
            FeatureDetector tempa = (FeatureDetector)input.get(a);
            if ((tempa!=null)&(tempa.inputCode==v)){
                vlines.add(tempa);
            } // end if tempa is horizontal line
        } // end for a copy list

        // assume segments are orphans (1) unless proved otherwise
        // note that this status does NOT affect whether segment can participate in other
pairings
        int[] status = new int[vlines.size()];
        for (int b=0; b<vlines.size(); b++) {
            status[b] = 1;
        } // end for b initialize status index

        for (int i=0; i<vlines.size(); i++) {
            FeatureDetector tempi = (FeatureDetector)vlines.get(i);
            for (int j=0; j<vlines.size(); j++) {
                  // this clause ensures that we don't compare an entry to itself (i=j)
                  // and that we don't check pairs ji that have already been checked as ij
                  if(j>i) {
                     FeatureDetector tempj = (FeatureDetector)vlines.get(j);
                     // must be at same x coordinates, but distinct y coordinates
                     if ((tempi.x==tempj.x)&(tempi.y!=tempj.y)){
                           // centers must be within 3 bits on y-axis (i.e, edges touching)
                           int centerDistance = Math.abs(tempi.y - tempj.y);
                           if (centerDistance<4) {
                              // ordering of segments on y-axis is handled by constructor
                              VerticalLine newLine = new VerticalLine(tempi, tempj);
                              output.add(newLine);
                              verticalLineObjectList.add(newLine);
                              //record that these line segments have been used at least once to
create larger line objects
                              status[i]=0;
                              status[j]=0;
                           } // end if close enough
                     } // end compare
                  } // end if j > i
            } // end for j
         } // end for i
         for (int k=0; k<vlines.size(); k++) {
            FeatureDetector tempk = (FeatureDetector)vlines.get(k);
            if(status[k]==1) {
                  VerticalLine orphanLine = new VerticalLine(tempk);
                  output.add(orphanLine);
                  // need vertical line object list?
            } // end if single
         } // end for k get singles

         return output;
   } // end consVerticalFeatures
   //--------

   // if line objects are in same y location, and edges touch, combine them into one larger
line object
   // use status index to keep track of any that are not used, and add them to list as singles
   // variable goon means go on
   public LinkedList consVerticalLineObjects (LinkedList input) {
         boolean goon = true;
         LinkedList output = new LinkedList();
         LinkedList copy = new LinkedList();
         for (int a=0; a<input.size(); a++) {
```

```
         VerticalLine tempa = (VerticalLine)input.get(a);
         copy.add(tempa);
      } // end for a copy list

      // 1 indicates need to include it as a single
      int[] status = new int[copy.size()];
      for (int b=0; b<copy.size(); b++) {
         status[b] = 1;
      } // end for b initialize status index

      for (int i=0; i<copy.size(); i++) {
         VerticalLine tempi = (VerticalLine)copy.get(i);
         for (int j=0; j<copy.size(); j++) {
               // this clause ensures that we don't compare an entry to itself (i=j)
               // and that we don't check pairs ji that have already been checked as ij
               if(j>i){
                  VerticalLine tempj = (VerticalLine)copy.get(j);
                  // lines must share same x-coordinate
                  //ignore lines with a common endpoint,a.ymin=b.ymin, or
a.ymax=b.ymax :  larger line covers it
                  //ignore lines subsumed within the other line, a.ymin<b.ymin AND
a.ymax>b.ymax :  a covers it
                  if (tempi.x==tempj.x) {
                        goon = true;
                        if ((tempi.ymin==tempj.ymin)|(tempi.ymax==tempj.ymax)) {
                           goon = false;
                        } // end if lines share an endpoint
                        if ((tempi.ymin<tempj.ymin)&(tempi.ymax>tempj.ymax)) {
                           goon = false;
                        } // end if a contains b
                        if ((tempj.ymin<tempi.ymin)&(tempj.ymax>tempi.ymax)) {
                           goon = false;
                        } // end if b contains a

                        // calculate gap from left-max to right-min
                        // adjacent lines will have distance = 1 (e.g., left-max = 3, right-
min = 4)
                        if (goon) {
                           int distance = 100;
                           if (tempi.ymin<tempj.ymin) {
                                 distance = tempj.ymin - tempi.ymax;
                           } // end if i..j
                           if (tempj.ymin<tempi.ymin) {
                                 distance = tempi.ymin - tempj.ymax;
                           } // end if j..i
                           // combine lines, set status for each part to 0
```

```
                    if (distance<2) {
                          VerticalLine newLine = new VerticalLine(tempi, tempj);
                          verticalLineObjectList.add(newLine);
                          output.add(newLine);
                          status[i]=0;
                          status[j]=0;
                    } // end if adjacent
                 } // end if goon
              }// end if same y location
           } // end if j > i
        } // end for j
     } // end for i

     for (int k=0; k<copy.size(); k++) {
        VerticalLine tempk = (VerticalLine)copy.get(k);
        if (status[k]==1) {
              output.add(tempk);
        } // end if single
     } // end for k get singles
     return output;
} // end consVerticalLineObjects
//---------

public LinkedList removeRedundantVerticalLines (LinkedList input) {
     LinkedList output = new LinkedList();
     LinkedList copy = new LinkedList();
     for (int a=0; a<input.size(); a++) {
        VerticalLine tempa = (VerticalLine)input.get(a);
        copy.add(tempa);
     } // end for a copy list
     // 0 indicates a duplicate
     int[] status = new int[copy.size()];
     for (int b=0; b<copy.size(); b++) {
        status[b] = 1;
     } // end for b initialize status index
     for (int i=0; i<copy.size(); i++) {
        VerticalLine tempi = (VerticalLine)copy.get(i);
        for (int j=0; j<copy.size(); j++) {
              // this clause ensures that we don't compare an entry to itself (i=j)
              // and that we don't check pairs ji that have already been checked as ij
              if(j>i) {
                 VerticalLine tempj = (VerticalLine)copy.get(j);
                 // must be in same x location
                 if (tempi.x==tempj.x) {
                       // if same location, arbitrarily remove second entry
                       // if tempi contains tempj, remove tempj
```

```
                        // if tempj contains tempi, remove tempi
                        if
((tempi.x==tempj.x)&(tempi.ymin==tempj.ymin)&(tempi.ymax==tempj.ymax)){
                           status[j]=0;
                        } else if
((tempi.ymin<=tempj.ymin)&(tempi.ymax>=tempj.ymax)) {
                           status[j]=0;
                        } else if
((tempj.ymin<=tempi.ymin)&(tempj.ymax>=tempi.ymax)) {
                           status[i]=0;
                        } // end if else else
                    } // end if x = x
                  } // end if j > i
              } // end for j
           } // end for i

           // put in output list any lines which have not been declared duplicates
           for (int k=0; k<copy.size(); k++) {
              VerticalLine tempk = (VerticalLine)copy.get(k);
              if (status[k]==1) {
                   output.add(tempk);
              } // end if single
           } // end for k get singles
           return output;
     } // end removeRedundantVerticalLines
     //----

     // methods to input Paint files and extract bitmaps
     public LinkedList bytesEcho(String infile) {
           LinkedList wordBytes = new LinkedList();
           try {
              FileInputStream file = new FileInputStream(infile);
              BufferedInputStream buff = new BufferedInputStream(file);
              boolean eof = false;
              while (!eof) {
                   int input = buff.read();
                   Integer bigInput = new Integer(input);
                   wordBytes.add(bigInput);
                   if (input == -1) {
                      eof = true;
                   } // end if eof
              } // end while
              buff.close();
           } catch (IOException e) {
           } // end catch
           return wordBytes;
```

```
} // end bytesEcho

// get height and width information
// height is 23rd character
int getHeight (LinkedList input) {
      int h = 0;
      Integer bigh = new Integer(0);
      bigh = (Integer)input.get(22);
      h = bigh.intValue();
      return h;
} // end getHeight

  // width is 19th character
int getWidth (LinkedList input) {
      int w = 0;
      Integer bigw = new Integer(0);
      bigw = (Integer)input.get(18);
      w = bigw.intValue();
      return w;
} // end get Width

// extract bitmap from bmp file
LinkedList bitTruncate(LinkedList input) {
      LinkedList output = new LinkedList();
      boolean bits = false;
      int[] whitecode = {255, 255, 255, 0};
      int[] temp = {-1, -1, -1, -1};
      // truncate everything up to and including 255 255 255 0
      for (int i=0; i< input.size(); i++) {
         Integer bigInt =(Integer)input.get(i);
         int in =bigInt.intValue();
         if (bits & (in!=-1)) {
               output.add(bigInt);
         } else {
               temp[0]=temp[1];
               temp[1] = temp[2];
               temp[2]=temp[3];
               temp[3] = in;
               if (temp[0]==whitecode[0] &
            temp[1]==whitecode[1] &
            temp[2]==whitecode[2] &
            temp[3]==whitecode[3]) {
                  bits = true;
               } // end if whitecode
         } // end not bits yet
      } // end for i
```

```
        return output;
    } // end bitTruncate

    public LinkedList binaryBits(LinkedList input) {
        LinkedList newbits = new LinkedList();
        // convert input digit into 8-bit binary code
        for (int i=0; i<input.size(); i++) {
          LinkedList bin = new LinkedList();
          Integer inputi = (Integer)input.get(i);
          int littleinputi = inputi.intValue();
          String binint = new String();
          int[] tempbits = {};
          Integer zero = new Integer(0);
          binint = Integer.toBinaryString(littleinputi);
          // convert to individual chars, to Strings, to ints
          // need to pad left: done in stringToBits
          tempbits = stringToBits(binint);
          for (int n=0; n<tempbits.length; n++) {
                int tempbitsn = tempbits[n];
                Integer bigtempbitsn = new Integer(tempbitsn);
                newbits.add(bigtempbitsn);
          } // end for n
        } // end for i
        return newbits;
    } // end binaryBits

    public int[] stringToBits (String bitstring) {
        Integer zero = new Integer(0);
        int max = bitstring.length();
        int[] bits = new int[max];
        int[] padbits = new int[8];
        for (int i=0; i<max; i++) {
          char temp = bitstring.charAt(i);
          String temps = temp + "";
          int tempint = Integer.parseInt(temps);
          bits[i] = tempint;
        } // end for i
        int diff = 8-bits.length;
        // pad left of padbits if necessary to make 8 digits
        // if already 8 digits, diff = 0, d<diff will be false
        for (int d=0; d<diff; d++) {
          //padbits.addFirst(zero);
          padbits[d]=0;
        } // end for d
        // enter digits from bits into padbits, to the right of padded zeroes
        for (int e=0; e<max; e++) {
            int f = diff + e;
            padbits[f] = bits[e];
        } // end for e
        return padbits;
    } // end stringToBits

    // extract bits returns a list of rows, representing the original picture in bits
    // white = 1, black =0
    public LinkedList extractBits(LinkedList input) {
        LinkedList output = new LinkedList();
        LinkedList bitoutput = new LinkedList();
        LinkedList rows = new LinkedList();
        // first get height and width information
        int h = getHeight(input);
        int w = getWidth(input);
        // then truncate header
        output = bitTruncate(input);
        // then convert to binary
        bitoutput = binaryBits(output);
        // then assemble based on height and width
        // each padded row = 32 bits
        // get a row and input only first w entries
        int rownumber = bitoutput.size()/32;
        for (int dummy=0; dummy<rownumber; dummy++) {
            LinkedList temprow = new LinkedList();
            for (int m=0; m<32; m++) {
                Integer bitoutputm = (Integer)bitoutput.get(m);
                temprow.add(bitoutputm);
            } // end for m
            // now remove 32 from bitoutput
            for (int p=0; p<32; p++) {
                bitoutput.removeFirst();
            }  // end for p remove 32
            int wdiff = 32 - w;
            for (int r=0; r<wdiff; r++) {
                temprow.removeLast();
            } // end for r remove right padding
            rows.addFirst(temprow);
        } // end for dummy
        return rows;
    } // end extractBits

    // bit grid matrix is 20 x 20
    public void clearBitGridMatrix() {
      for (int i=0; i<20; i++) {
            for (int j=0; j<20; j++) {
                bitGridMatrix[i][j] = null;
            } // end for j
        } // end for i
    } // end clearBitGridMatrix

    // given input in row format from extractBits, compute 3x3 bit grids
    // puts these into bitGridMatrix as a side-effect
    // note:  avoid edges
    public void bitGrids(LinkedList input) {
        clearBitGridMatrix();
        int h = getHeight(input);
        int w = getWidth(input);
        LinkedList rows = extractBits(input);
        for (int x=1; x<w-1; x++) {
          for(int y=1; y<h-1; y++) {
                BitGrid bgxy = new BitGrid(x,y);
                LinkedList rowa = (LinkedList)rows.get(y-1);
                LinkedList rowb = (LinkedList)rows.get(y);
                LinkedList rowc = (LinkedList)rows.get(y+1);

                Integer bigInt0 = (Integer)rowa.get(x-1);
                int int0 = bigInt0.intValue();
                bgxy.input[0] = int0;

                Integer bigInt1 = (Integer)rowa.get(x);
                int int1 = bigInt1.intValue();
                bgxy.input[1] = int1;

                Integer bigInt2 = (Integer)rowa.get(x+1);
                int int2 = bigInt2.intValue();
                bgxy.input[2] = int2;

                Integer bigInt3 = (Integer)rowb.get(x-1);
                int int3 = bigInt3.intValue();
                bgxy.input[3] = int3;

                Integer bigInt4 = (Integer)rowb.get(x);
                int int4 = bigInt4.intValue();
                bgxy.input[4] = int4;

                Integer bigInt5 = (Integer)rowb.get(x+1);
                int int5 = bigInt5.intValue();
                bgxy.input[5] = int5;

                Integer bigInt6 = (Integer)rowc.get(x-1);
                int int6 = bigInt6.intValue();
                bgxy.input[6] = int6;

                Integer bigInt7 = (Integer)rowc.get(x);
                int int7 = bigInt7.intValue();
                bgxy.input[7] = int7;

                Integer bigInt8 = (Integer)rowc.get(x+1);
                int int8 = bigInt8.intValue();
                bgxy.input[8] = int8;

                bitGridMatrix[x][y] = bgxy;
          } // end for x
       }// end for y
} // end bitGrids
//--------------------------------------------------------

public void printBitGridMatrix () {
       System.out.println("bitGridMatrix");
       for (int x =0; x<20; x++) {
          System.out.print("row " + x + ": ");
          for (int y=0; y<20; y++) {
                if (!(bitGridMatrix[x][y]==null)) {
                   BitGrid temp = bitGridMatrix[x][y];
                   System.out.print(temp.x + "," + temp.y + " ");
                } // end if not null
          } // end for y
          System.out.println();
       } // end for x
} // end printBitGridMatrix

public void printFeatures() {
       System.out.println("order of feature detectors (using 1,1 as example)");
    FeatureDetector[] firstList = featureListMatrix[1][1];
       if (!(firstList==null)) {
          for (int f=0; f<firstList.length; f++) {
                FeatureDetector fdf = firstList[f];
                if (!(fdf==null)) {
                   System.out.println(fdf.description);
                } // end if
          } // end for f
       } // if not null
} // end print features
//------------
public void initiateHorizontalLineDetectors() {
       //1
       FeatureDetector fdh1$1 = new FeatureDetector(h, 1, 1, "horizontal line at 1,1");
```

```
        FeatureDetector fdh1$2 = new FeatureDetector(h, 1, 2, "horizontal line at 1,2");
        FeatureDetector fdh1$3 = new FeatureDetector(h, 1, 3, "horizontal line at 1,3");
        FeatureDetector fdh1$4 = new FeatureDetector(h, 1, 4, "horizontal line at 1,4");
        FeatureDetector fdh1$5 = new FeatureDetector(h, 1, 5, "horizontal line at 1,5");
        FeatureDetector fdh1$6 = new FeatureDetector(h, 1, 6, "horizontal line at 1,6");
        FeatureDetector fdh1$7 = new FeatureDetector(h, 1, 7, "horizontal line at 1,7");
        FeatureDetector fdh1$8 = new FeatureDetector(h, 1, 8, "horizontal line at 1,8");
        FeatureDetector fdh1$9 = new FeatureDetector(h, 1, 9, "horizontal line at 1,9");
....
//------------
  // decay of activation

  // feature-level activation decays quickly; object-level activation decays moderately
  // features are stored in feature lists, which are stored in featureListMatrix[x][y][input]
  // objects are stored in masterObjectList
  // for implementation, only calculate decay for things which are active above a certain
threshold
  // keep floor above decay amount to avoid negative activation strengths

  public void decayFeatureActivation(double time) {
        System.out.println("decaying feature activation at time " + time);
        for (int i=0; i<20; i++) {
          for (int j=0; j<20; j++) {
                for (int k=0; k<40; k++) {
                  FeatureDetector tempf = featureListMatrix[i][j][k];
                  if (tempf != null) {
                        if (tempf.act>0) {
                          // System.out.println("act is " + tempf.act + " for " +
tempf.description);
                        } // end comment
                        if (tempf.act>featureDecayFloor) {
                          tempf.act = tempf.act - featureDecayAmount;
                          //  System.out.println("decaying activation of " +
tempf.description);
                          //System.out.println("to new value of " + tempf.act);
                        } // end if above floor
                  } // end if screen nulls
                } // end for k inputCode
          } // end for j yloc
        } // end for i xloc
  } // end decayFeatureActivation

  // keep floor > decay amount to avoid negative activation strengths
  public void decayObjectActivation(double time) {
        System.out.println("decaying object activation at time " + time);
        for (int i=0; i<masterObjectList.size(); i++) {
            Percept percepti = (Percept)masterObjectList.get(i);
            if (percepti.act>objectDecayFloor) {
                  percepti.act = percepti.act - objectDecayAmount;
                  System.out.println("decaying object activation to " + percepti.act);
            } // end if above floor
        } // end for i master object list
    } // end decayObjectActivation

    // keep floor > decay amount to avoid negative activation strengths
    public void decayConceptActivation(double time) {
        System.out.println("decaying concept activation at time " + time);
        for (int i=0; i<masterConceptList.size(); i++) {
            Concept concepti = (Concept)masterConceptList.get(i);
            if (concepti.act>conceptDecayFloor) {
                  System.out.println("original amount is " + concepti.act);
                  concepti.act = concepti.act - conceptDecayAmount;
                  System.out.println("decaying concept activation to " + concepti.act);
            } // end if above floor
        } // end for i master concept list
    } // end decayConceptActivation

    //------------

    // look at all feature lists, check xloc yloc to match bitgrid xloc yloc
    // then, if location matches, check all feature detectors in that list
    // shield null entries:  note that array length is total length of openings, not length of
entries
    public void bitGridActivation(double time) {
        for(int x=0; x<20; x++) {
            for(int y=0; y<20; y++) {
                  BitGrid bgxy = bitGridMatrix[x][y];
                  if(!(bgxy==null)) {
                      FeatureDetector[] flistxy = featureListMatrix[x][y];
                      if (!(flistxy==null)) {
                            for (int f=0; f<flistxy.length; f++) {
                                FeatureDetector featureDetectorf = flistxy[f];
                                if (!(featureDetectorf==null)) {
                                      int partialActivation = compareVectors(bgxy.input,
featureDetectorf.inputCode);
                                      double proportionalActivation = partialActivation / 9.0;
                                      if
(proportionalActivation>featureImplementationThreshold) {
                                          featureDetectorf.act = featureDetectorf.act +
featureInputWeight*proportionalActivation;
                                      } // end if over thrshold
                                } // end if not null feature detector
```

```
                        } // end for f
                    } // end if not null feature list
                  } // end if not null bitgrid
              } // end for y
          } // end for x
    } // end bitGridActivation

    //-----------
    // checks objects on masterObjectList, and slices on object activation
    // if activation is high enough, fires, and is then inhibited for a time
    // because of this timing issue, getActiveObjects should only be called by
processObjects
    public LinkedList getActiveObjects(double time) {
          printMasterObjectList();
          // first, reset activeObjects list
          // percept.fire will add objects to activeObjects list
          activeObjects = new LinkedList();
          for (int i=0; i<masterObjectList.size(); i++) {
             Percept percepti = (Percept)masterObjectList.get(i);
             percepti.checkInhibition(time);
             if ((percepti.act>objectThreshold)&(!percepti.off)) {
                   percepti.fire(time);
             } // end if
          } // end for i all objects
          return activeObjects;
    } // end getActiveObjects

    // currently, only squares have lateral associations
    public void lateralObjectActivation(double time) {
          // first top-down spreading to component objects
          LinkedList allDescendants = new LinkedList();
          for (int i=0; i<activeObjects.size(); i++) {
             Percept percepti = (Percept)activeObjects.get(i);
             LinkedList descendants = new LinkedList();
             //get type and cast to type
             //get parts
             //send activation to component objects

             System.out.println("lateral activation: percepti.type " + percepti.type);
             if (percepti.type=="square") {
                   Square sq = (Square)percepti;
                   LinkedList objects = sq.componentObjects;
                   for (int j=0; j<objects.size(); j++) {
                      Percept perceptj = (Percept)objects.get(j);
                      activateObject(perceptj, lateralObjectActivationIncrement);
                   } // end for j objects
```

```
            } // end for i square
        } // end for i percepts

        // next, bottom-up activation of containing objects
        // do not check for newly active objects based on top-down activation yet
        // checkSquares method uses activeObjects list
        checkSquares();
    } // end lateralObjectActivation

    //-----------
    // method fire puts features on global variable, activeFeatures
    public LinkedList getActiveFeatures(double time) {
        // first, reset activeFeatures list
        // method FeatureDetector.fire will add feature detectors to the list, activeFeatures
        activeFeatures = new LinkedList();
        for(int x=0; x<20; x++) {
          for(int y=0; y<20; y++) {
                for (int z=0; z<40; z++) {
                  FeatureDetector fff = featureListMatrix[x][y][z];
                  if (fff!=null) {
                        fff.checkInhibition(time);
                        if ((fff.act>featureThreshold)&(!fff.off)) {
                          fff.fire(time);
                          // activeFeatures.add(fff);
                        } // end if over threshold
                  } // end if not null
                } // end for z
          } // end for y
        } // end for x
        return activeFeatures;
    } // end getActiveFeatures
    //------
    public void detectHorizontalLines(LinkedList features) {
        LinkedList basicLines = consHorizontalFeatures(features);
        LinkedList derivedLines = consHorizontalLineObjects(basicLines);
        LinkedList nonredundantLines =
removeRedundantHorizontalLines(derivedLines);
        LinkedList extendedLines = extendHorizontalLines(nonredundantLines);
        LinkedList consExtLines = consHorizontalLineObjects(extendedLines);
        LinkedList finalLines = removeRedundantHorizontalLines(consExtLines);
        for (int i=0; i<finalLines.size(); i++) {
          Percept percepti = (Percept)finalLines.get(i);
          activateObject(percepti, objectActivationWeight);
        } // end for i final lines
    } // end detect horizontal
```

```
  public void detectVerticalLines(LinkedList features) {
        LinkedList basicLines = consVerticalFeatures(features);
        LinkedList derivedLines = consVerticalLineObjects(basicLines);
        LinkedList nonredundantLines = removeRedundantVerticalLines(derivedLines);
        LinkedList extendedLines = extendVerticalLines(nonredundantLines);
        LinkedList consExtLines = consVerticalLineObjects(extendedLines);
        LinkedList finalLines = removeRedundantVerticalLines(consExtLines);
        for (int i=0; i<finalLines.size(); i++) {
           Percept percepti = (Percept)finalLines.get(i);
           activateObject(percepti, objectActivationWeight);
        } // end for i final lines
  } // end detect vertical

  public void detectDiagonalLinesSWNE(LinkedList features) {
        LinkedList basicLines = consDiagonalSWNEFeatures(features);
        LinkedList derivedLines = consDiagonalSWNELineObjects(basicLines);
        LinkedList finalLines = removeRedundantDiagonalSWNELines(derivedLines);
        for (int i=0; i<finalLines.size(); i++) {
           Percept percepti = (Percept)finalLines.get(i);
           activateObject(percepti, objectActivationWeight);
        } // end for i final lines
  } // end detect diag swne

  public void detectDiagonalLinesNWSE(LinkedList features) {
        LinkedList basicLines = consDiagonalNWSEFeatures(features);
        LinkedList derivedLines = consDiagonalNWSELineObjects(basicLines);
        LinkedList finalLines = removeRedundantDiagonalNWSELines(derivedLines);
        for (int i=0; i<finalLines.size(); i++) {
           Percept percepti = (Percept)finalLines.get(i);
           activateObject(percepti, objectActivationWeight);
        } // end for i final lines
  } // end detect diag nwse

  //-----------
  // based on features, detect objects
  // note: for square of size 3, must recognize directly from features, not objects
  // since the bit grids for the corners and sides overlap
  public void spreadActivationFromFeaturesToObjects(double time, LinkedList
features) {
        consolidateHorizontalLines(features);
        consolidateVerticalLines(features);
        consolidateDiagonalSWNELines(features);
        consolidateDiagonalNWSELines(features);
        checkCorners(features);
        checkCorners3D(features);
        checkPartsSquare3(features);
```

```
} // end spreadActivationFromFeaturesToObjects
//----------
// activateObject compares percept to existing percepts based on type, location, size
// if not already on list, add to list and use actinc for initial activation
// if already on list, modify existing activation by adding actinc
// should be called with proportionMatched*objectActivationWeight
// if object is certain, use implicit 1.0 * objectActivationWeight
public void activateObject(Percept perc, double actinc) {
      boolean newObject = true;
      for (int i=0; i<masterObjectList.size(); i++) {
         Percept percepti = (Percept)masterObjectList.get(i);
         if
((percepti.type==perc.type)&(percepti.x==perc.x)&(percepti.y==perc.y)&(percepti.size=
=perc.size)) {
               newObject = false;
               percepti.act = percepti.act + actinc;
         } // end if matching

         // check for lines contained in lines
         // if new object is a line contained in an existing line, propagate the line
segment,
         // and copy the activation of the existing line
         if ((newObject)&(percepti.type==perc.type)&(perc.type=="lineh")) {
               // cast to hline
               HorizontalLine nowH = (HorizontalLine)percepti;
               HorizontalLine soughtH = (HorizontalLine)perc;
               // if line contained in existing line, copy activation
               // new object will be created below
               if
((nowH.y==soughtH.y)&(nowH.xmin<=soughtH.xmin)&(nowH.xmax>=soughtH.xmax)
){
                  perc.act = percepti.act;
                  //System.out.println("line needed is contained in existing line");
                  //System.out.println("short line copies act: " + perc.act);
               } // end if matches
         } // end if hline
         if ((newObject)&(percepti.type==perc.type)&(perc.type=="linev")) {
               // cast to hline
               VerticalLine nowV = (VerticalLine)percepti;
               VerticalLine soughtV = (VerticalLine)perc;
               if
((nowV.x==soughtV.x)&(nowV.ymin<=soughtV.ymin)&(nowV.ymax>=soughtV.ymax)
){
                  perc.act = percepti.act;
                  //System.out.println("line needed is contained in existing line");
                  //System.out.println("short line copies act: " + perc.act);
```

```
                } // end if matches
            } // end if vline
        } // end for i master object list

        if (newObject) {
            perc.act = perc.act + actinc;
            masterObjectList.add(perc);
            //System.out.println("act becomes " + perc.act);
        }
    } // end activateObject

    // ---
    public void activateConcept(Concept con, double actinc) {
        boolean newConcept = true;
        for (int i=0; i<masterConceptList.size(); i++) {
            Concept concepti = (Concept)masterConceptList.get(i);
            if
((concepti.name==con.name)&(concepti.x==con.x)&(concepti.y==con.y)&(concepti.size
==con.size)){
                newConcept = false;
                concepti.act = concepti.act + actinc;
            } // end if existing concept
        } // end for i master concept list
        if (newConcept) {
            con.act = con.act + actinc;
            masterConceptList.add(con);
        } // end if new concept
    } // end activateConcept

    //----------
    // shift attention to one 3D corner
    // should favor one interpretation of the Necker Cube (CubeA vs. CubeB, here)
    // this method gives attention to any corner sw 3d -- not a perfect simulation of fixating
one corner
    public void attendToCornerSW3D(double time) {
        System.out.println("at attend sw at time " + time);
        LinkedList corners = new LinkedList();
        for (int i=0; i<masterObjectList.size(); i++) {
            Percept percepti = (Percept)masterObjectList.get(i);
            if (percepti.type=="cornersw3d") {
                CornerSW3D corneri = (CornerSW3D)percepti;
                corners.add(corneri);
            } // end if corner sw 3d
        } // end for i master object list
        // add activation to each corner sw 3d on list
        for (int j=0; j<corners.size(); j++) {
            CornerSW3D cornerj = (CornerSW3D)corners.get(j);
            activateObject(cornerj, 1.0);
            System.out.println("increased activation of corner sw 3d at " + cornerj.x + "," +
cornerj.y + " to " + cornerj.act);
        } // end for j corners
    } // end attendToCornerSW3D
    //----------
    public void attendToCornerNE3D(double time) {
        System.out.println("at attend ne at time " + time);
        LinkedList corners = new LinkedList();
        for (int i=0; i<masterObjectList.size(); i++) {
            Percept percepti = (Percept)masterObjectList.get(i);
            if (percepti.type=="cornerne3d") {
                CornerNE3D corneri = (CornerNE3D)percepti;
                corners.add(corneri);
            } // end if corner ne 3d
        } // end for i master object list
        // add activation to each corner ne 3d on list
        for (int j=0; j<corners.size(); j++) {
            CornerNE3D cornerj = (CornerNE3D)corners.get(j);
            activateObject(cornerj, 1.0);
            System.out.println("increased activation of corner ne 3d at " + cornerj.x + "," +
cornerj.y + " to " + cornerj.act);
        } // end for j corners
    } // end attendToCornerNE3D
    //----------

    public static LinkedList deriveShortList (LinkedList fullLL) {
        LinkedList shortLL = new LinkedList();
        for (int i=0; i<fullLL.size(); i++) {
            if (!shortLL.contains(fullLL.get(i)) ) {
                shortLL.add(fullLL.get(i));
            } // end if
        } // end for i
        return (shortLL);
    } // end deriveShortList

    //-----------

    // this function takes one function and crosses it by itself
    // exclude i=j, and only check pair once: having checked 0x1, don't check 1x0
    // use this function to build other functions; add new code in "found match" section
    public void pairwiseCross (LinkedList input) {
        LinkedList copy = new LinkedList();
        for (int a=0; a<input.size(); a++) {
            String tempa = (String)input.get(a);
            copy.add(tempa);
        } // end for a copy list
        for (int i=0; i<copy.size(); i++) {
            String tempi = (String)copy.get(i);
            for (int j=0; j<copy.size(); j++) {
                // this clause ensures that we don't compare an entry to itself (i=j)
                // and that we don't check pairs ji that have already been checked as ij
                if(j>i) {
                    String tempj = (String)copy.get(j);
                    if(tempi==tempj) {
                        //System.out.println("found match " + i + ":" + tempi + "=" + j +
":" + tempj);
                    } // end compare
                } // end if j > i
            } // end for j
        } // end for i
    } // end pairwiseCross

    // remove redundant entries from lists of strings
    // for primitive entries, could use contains method,
    // but for complex objects or complex comparisons, use this
    // put comparison in place of tempi==tempj
    // e.g., tempi.xval == tempj.xval
    public LinkedList removeRedundantEntries(LinkedList input) {
        LinkedList output = new LinkedList();
        LinkedList copy = new LinkedList();
        String[] status = new String[input.size()];
        for (int a=0; a<input.size(); a++) {
            String tempa = (String)input.get(a);
            copy.add(tempa);
        } // end for a copy list
        for (int b=0; b<input.size(); b++) {
            status[b] = "OK";
        } // end for b status array
        for (int i=0; i<copy.size(); i++) {
            String tempi = (String)copy.get(i);
            if (status[i]=="OK") {
                status[i]="tested";
                for (int j=0; j<copy.size(); j++) {
                    String tempj = (String)copy.get(j);
                    // don't compare to self
                    if (i!=j) {
                        if (status[j]=="OK") {
                            //perform comparison
                            // setting status[j] to "duplicate" effectively removes tempj
                            if (tempi==tempj) {
```

```
                                        status[j]="duplicate";
                                   } // end if matched
                              } // end if status j OK
                         } // end for j
                    } // end if i != j
               }
          } // end for i

          /*
            System.out.println("status index ");
            for (int p=0; p<status.length; p++) {
            System.out.print(" " + copy.get(p) + ":" + status[p]);
            } // end for p print status
            System.out.println("");
          */

          for (int k=0; k<copy.size(); k++) {
             String tempk = (String)copy.get(k);
             if (status[k]!="duplicate") {
                  output.add(tempk);
             } // status "OK" or "tested" is acceptable
          } // end for k compile new list
          return output;
     } // end removeRedundantEntries

     //==================

     public void printArray(int[] a) {
          System.out.print("[ ");
          for (int i=0; i< a.length; i++) {
             System.out.print(a[i] + " ");
          } // end for i
          System.out.println("]");
     } // end printArray

     public void printBitGrid(BitGrid b) {
          int[] a = b.input;
          System.out.println("-- bit grid --");
          System.out.print(a[0] + " " + a[1] + " " + a[2]);
          System.out.println();
          System.out.print(a[3] + " " + a[4] + " " + a[5]);
          System.out.println();
          System.out.print(a[6] + " " + a[7] + " " + a[8]);
          System.out.println();
     } // end printBitGrid
```

```
// utility to check two binary inputs for consistency
// must contain enough 0 (black) elements to contain target
// may contain additional 0 (black) elements, taken to represent
// an additional feature overlaying the target feature
public double compareWithOverlay(int[] target, int[] poss) {
        int total = target.length;
        double proportion = 0.0;
        for (int i=0; i<target.length; i++) {
           if ((target[i]==0)&(poss[i]==1)){
                total = total - 1;
           } // end if poss has white where target has black
        } // end for i bits
        proportion = total / target.length;
        //System.out.println("compareWithOverlay finds " + total + " consistent bits");
        return proportion;
} // end compareWithOverlay

// utility to check two binary inputs for matches
// if both are 1, a&b is true
// if both are 0, a|b is false
public boolean bitmatch(int a, int b) {
        boolean match = false;
        if (((a&b)==1)|((a|b)==0)) {
           match = true;
        } // end if
        return match;
} // end bitmatch

// compare two binary vectors
// vectors must match in length
// uses utility, bitmatch
public int compareVectors(int[] inputa, int[] inputb) {
        int total = 0;
        for (int i=0; i<inputa.length; i++) {
           int a = inputa[i];
           int b = inputb[i];
           if ( bitmatch(a, b) ) {
                total = total + 1;
           } // end if bitmatch
        }  // end for i
        ///System.out.println("total number of matches is " + total);
           return total;
} // end compareVectors

// ---
public void printMasterObjectList(){
        System.out.println("master object list ");
        for (int z=0; z<masterObjectList.size(); z++) {
           Percept tempz = (Percept)masterObjectList.get(z);
           System.out.println(tempz + " " + tempz.type +  " (" + tempz.x + "," + tempz.y +
") act: " + tempz.act);
        } // end for z master object list
   } // end printMasterObjectList

   // ---
   public void printMasterConceptList(){
        System.out.println("master concept list " + masterConceptList);
        for (int z=0; z<masterConceptList.size(); z++) {
           Concept tempz = (Concept)masterConceptList.get(z);
           System.out.println(tempz + " " + tempz.name +  " (" + tempz.x + "," + tempz.y
+ ") act: " + tempz.act + " off =" + tempz.off);
        } // end for z master concept list
   } // end printMasterConceptList
   //---------
   public void printActiveConcepts(){
        System.out.println("active concepts are " + activeConcepts);
        for (int z=0; z<activeConcepts.size(); z++) {
           Concept tempz = (Concept)activeConcepts.get(z);
           System.out.println(tempz + " " + tempz.name +  " (" + tempz.x + "," + tempz.y
+ ") act: " + tempz.act);
        } // end for z active concept list
   } // end printActiveConcepts
   //---------
   public void printActiveObjects(){
        System.out.println("active objects are ");
        for (int z=0; z<activeObjects.size(); z++) {
           Percept tempz = (Percept)activeObjects.get(z);
           System.out.println(tempz + " " + tempz.type +  " (" + tempz.x + "," + tempz.y +
") act: " + tempz.act);
        } // end for z active object list
   } // end printActiveObjects

   //---------

   // using modulus operator to get remainder
   public void updateTime(double time) {
        double timePassed = time - oldTime;
        oldTime = time;  // for next cycle
        featureTime = featureTime + timePassed;
        objectTime = objectTime + timePassed;
        conceptTime = conceptTime + timePassed;
        //System.out.println("time: " + time + ", time passed: " + timePassed);
        //System.out.println("feature time updated to " + featureTime);
        //System.out.println("object time updated to " + objectTime);

        if (featureTime>=featureDelta) {
           // reset time for next cycle using modulus
           featureTime = featureTime % featureDelta;
           //System.out.println("feature time reset to " + featureTime);
        } // end if need to update feature cycle

        if (objectTime>=objectDelta) {
           // reset time for next cycle
           // using modulus operator which gives remainder after division
           objectTime = objectTime % objectDelta;
           //System.out.println("object time reset to " + objectTime);
        } // end if

        if (conceptTime>=conceptDelta) {
           // reset time for next cycle
           // using modulus operator which gives remainder after division
           conceptTime = conceptTime % objectDelta;
           //System.out.println("concept time reset to " + conceptTime);
        } // end if
    } // end updateTime
    //---------

    public void spreadActivationFromObjectsToFeatures(double time) {
        //System.out.println("at spreadAct from obj to features");
        LinkedList features = new LinkedList();
        //System.out.println("active objects are " + activeObjects);
        LinkedList allDescendants = new LinkedList();
        for (int j=0; j<activeObjects.size(); j++) {
           Percept perceptj = (Percept)activeObjects.get(j);
           LinkedList descendants = new LinkedList();
           //get type and cast to type
           //get parts
           //send activation to feature detectors for parts
           //System.out.println("perceptj.type " + perceptj.type);
           if (perceptj.type=="lineh") {
                 HorizontalLine hline = (HorizontalLine)perceptj;
                 descendants = hline.getFD();
           } else if (perceptj.type=="linev") {
                 VerticalLine vline = (VerticalLine)perceptj;
                 descendants = vline.getFD();
           }else if (perceptj.type=="diagswne") {
                 DiagonalLineSWNE dline1 = (DiagonalLineSWNE)perceptj;
                 descendants = dline1.getFD();
```

```
          }else if (perceptj.type=="diagnwse") {
               DiagonalLineNWSE dline2 = (DiagonalLineNWSE)perceptj;
               descendants = dline2.getFD();
          } else if (perceptj.type=="cornernw") {
               CornerNW cnw = (CornerNW)perceptj;
               descendants = cnw.getFD();
          } else if (perceptj.type=="cornersw") {
               CornerSW csw = (CornerSW)perceptj;
               descendants = csw.getFD();
          } else if (perceptj.type=="cornerne") {
               CornerNE cne = (CornerNE)perceptj;
               descendants = cne.getFD();
          } else if (perceptj.type=="cornerse") {
               CornerSE cse = (CornerSE)perceptj;
               descendants = cse.getFD();
          } else if (perceptj.type=="square3") {
               Square3 sq3 = (Square3)perceptj;
               descendants = sq3.getFD();
          } else if (perceptj.type=="square") {
               Square sq = (Square)perceptj;
               descendants = sq.getFD();
          } else if (perceptj.type=="cornerne3d") {
               CornerNE3D cne3d = (CornerNE3D)perceptj;
               descendants = cne3d.getFD();
          } else if (perceptj.type=="cornerse3d") {
               CornerSE3D cse3d = (CornerSE3D)perceptj;
               descendants = cse3d.getFD();
          } else if (perceptj.type=="cornersw3d") {
               CornerSW3D csw3d = (CornerSW3D)perceptj;
               descendants = csw3d.getFD();
          } else if (perceptj.type=="cornernw3d") {
               CornerNW3D cnw3d = (CornerNW3D)perceptj;
               descendants = cnw3d.getFD();
          } // end if else object type
          //System.out.println("descendants " + descendants);
          allDescendants.addAll(descendants);
     } // end for j active objects
     //System.out.println("all descendants "  + allDescendants);

     for (int d=0; d<allDescendants.size(); d++) {
          FeatureDetector featured = (FeatureDetector)allDescendants.get(d);
          //System.out.println("adding activation to " + featured.description + ", " +
featured.act);
          featured.act = featured.act +
objectBasedActivationIncrement*featureObjectWeight;
          //System.out.println("activation becomes " + featured.act);
```

```
        } // end for d allDescendants
    } // end spread objects to features
    //--------

    public void getActiveConcepts(double time) {
        //System.out.println("at getActiveConcepts");
        //System.out.println("masterConceptList has " + masterConceptList.size() + "
entries");
        activeConcepts = new LinkedList();
        for (int i=0; i<masterConceptList.size(); i++) {
            Concept concepti = (Concept)masterConceptList.get(i);
            //System.out.println("potential concept of type " + concepti.name);
            //System.out.println("with activation " + concepti.act);
            concepti.checkInhibition(time);
            //System.out.println("concepti.off " + concepti.off);
            if ((concepti.act>conceptThreshold)&(!concepti.off)) {
                //System.out.println("concept over threshold " + concepti.name + " at " +
concepti.x + "," + concepti.y);
                concepti.fire(time);
            } // end if
        } // end for i all objects
    } // end get active concepts

    // -----
    //top down activation from cubes to components
    public void spreadActivationFromConceptsToObjects(double time) {
        for (int i=0; i<activeConcepts.size(); i++) {
            Concept c = (Concept)activeConcepts.get(i);
            LinkedList objects = c.relatedObjects;
            //System.out.println("related objects are " + objects);
            for (int j=0; j<objects.size(); j++) {
                Percept perceptj = (Percept)objects.get(j);
                activateObject(perceptj, conceptToObjectActivationIncrement);
            } // end for j objects
        } // end for i concepts
    } // end spread concepts to objects
    //--------
    // if sufficient objects exist on activeObjects,
    // create concept and activate it to some extent
    public void spreadActivationFromObjectsToConcepts(double time) {
        //System.out.println("at spread activation from objects to concepts, at time " +
time);
        checkCubeA();
        checkCubeB();
    } // end spread objects to concepts
    //--------
```

```
  // feature activation comes from both bitGrid input and from active objects
  // bitGrid info stored in bitGridMatrix; object info stored in masterObjectList
  // note must screen for derived features in off-screen locations
  public void processFeatures(double time) {
        bitGridActivation(time);
        getActiveFeatures(time);
        spreadActivationFromFeaturesToObjects(time, activeFeatures);
  } // end processFeatures
  //-------
  // lateral : spread activation to other, related objects
  public void processObjects(double time) {
        getActiveObjects(time);
        spreadActivationFromObjectsToConcepts(time);
        lateralObjectActivation(time);
        spreadActivationFromObjectsToFeatures(time);
  } // end processObjects
  //-------

  // put concepts which are over threshold into a buffer, for later firing
  public LinkedList getActiveConceptsBuffer(double time) {
        LinkedList overThreshold = new LinkedList();
        System.out.println("master concept list " + masterConceptList);
        //        activeConcepts = new LinkedList();
        for (int i=0; i<masterConceptList.size(); i++) {
           Concept concepti = (Concept)masterConceptList.get(i);
           System.out.println("potential object of type " + concepti.name);
           System.out.println("with activation " + concepti.act);
           concepti.checkInhibition(time);
           System.out.println("concepti.off " + concepti.off);
           if ((concepti.act>conceptThreshold)&(!concepti.off)) {
                 System.out.println("concept over threshold " + concepti.name + " at " +
concepti.x + "," + concepti.y);
                 overThreshold.add(concepti);
           } // end if
        } // end for i all objects
        return overThreshold;
  } // end getActiveConceptsBuffer
  //-------
  // in more general use, would restrict this function to apply to concepts in the same
location
  // here, the only concepts are the two cube interpretations, so they are always in
conflict
  // select the interpretation with the higher level of activation
  // when activations are equal, let both concepts remain
  public LinkedList resolveBufferConflicts(double time, LinkedList buffer) {
        LinkedList winners = new LinkedList();
        System.out.println("time=" + time + ", at resolve buffer conflicts with " + buffer);
        LinkedList copy = new LinkedList();
        for (int a=0; a<buffer.size(); a++) {
           Concept tempa = (Concept)buffer.get(a);
           copy.add(tempa);
        } // end for a copy list
        for (int i=0; i<copy.size(); i++) {
           Concept tempi = (Concept)copy.get(i);
           for (int j=0; j<copy.size(); j++) {
                 // this clause ensures that we don't compare an entry to itself (i=j)
                 // and that we don't check pairs ji that have already been checked as ij
                 if(j>i) {
                    Concept tempj = (Concept)copy.get(j);
                    //System.out.println("::secondary concept is " + tempj.name);
                    if(((tempi.name=="cubeA")&(tempj.name=="cubeB"))|
                 ((tempi.name=="cubeB")&(tempj.name=="cubeA"))) {
                          System.out.println("conflict between " + tempi.name + " act " +
tempi.act + " and " + tempj.name + " act " + tempj.act);
                          if (tempi.act>tempj.act) {
                             System.out.println("removing " + tempj.name);
                             tempj.act = 0.0; // added inhibition
                             buffer.remove(tempj);
                          } else if (tempi.act<tempj.act) {
                             System.out.println("removing " + tempi.name);
                             tempi.act = 0.0; // added inhibition
                             buffer.remove(tempi);
                          } else if (tempi.act==tempj.act) {
                             // leave conflict
                             System.out.println("activations equal");
                          } // end if
                    } // end compare
                 } // end if j > i
           } // end for j
        } // end for i
        System.out.println("list of concepts over threshold becomes " + buffer);
        winners = buffer;
        return winners;
  } // end resolveBufferConflicts
  //-------
  // fire any concepts which remain on the buffer; call after calling
resolveBufferConflicts
  public void fireConcepts(double time, LinkedList buffer) {
        activeConcepts = new LinkedList();
        for (int i=0; i<buffer.size(); i++) {
           Concept ci = (Concept)buffer.get(i);
```

```
            ci.fire(time);
        } // end for i
    } // end fireConcepts
    //-------

    public void processConcepts(double time) {
        System.out.println("processing concepts at time " + time);
        printMasterConceptList();
        LinkedList bufferConcepts = getActiveConceptsBuffer(time);
        LinkedList fireBuffer = resolveBufferConflicts(time, bufferConcepts);
        fireConcepts(time, fireBuffer);
        spreadActivationFromConceptsToObjects(time);
    } // end processConcepts
    //-----

    public void processAll(double time, String file) {
        // first process input file
        LinkedList bits = bytesEcho(file);
        extractBits(bits);
        bitGrids(bits);

        // updateTime resets
        updateTime(time);
        if (time%featureDelta ==0) {
            decayFeatureActivation(time);
            processFeatures(time);
        } // end if features
        if (time%objectDelta ==0) {
            processObjects(time);
            decayObjectActivation(time);
        } // end if objects
        if (time%conceptDelta==0) {
            processConcepts(time);
            decayConceptActivation(time);
        } // end if concepts
        printActiveObjects();
    } // end processAll

    // --------
    public LinkedList createInputFileList() {
        LinkedList files = new LinkedList();
        files.add("a1.bmp");
        files.add("a2.bmp");
        files.add("a3.bmp");
        files.add("a4.bmp");
        files.add("a5.bmp");
        files.add("a6.bmp");
        files.add("a7.bmp");
        return files;
    } // end createInputFileList

    public void processFiles() {
        LinkedList fileList = createInputFileList();
        // can't process more input than there is, so max is set to input file list size
        int maxTime = 13;
        int max = fileList.size();
        if (maxTime<max) {
           max = maxTime;
        } // end if time set smaller than input file list size

        for (int t=1; t<max; t++) {
           String filet = (String)fileList.get(t);
           processAll(t, filet);
        } // end for time
    } // end processFiles
    //----------
    // attention
    // note to programmer: should pull object activation above loop
    //-- would matter if more than one concept linked to this object
    public void attendToObject(double time, Percept perc, double actinc) {
        //System.out.println("*& at attend to object with " + perc);
        for (int i=0; i<masterConceptList.size(); i++) {
           Concept concepti = (Concept)masterConceptList.get(i);
           //System.out.println(concepti + " relatedObjects are " +
concepti.relatedObjects);
           if (containsPercept(perc, concepti.relatedObjects)) {
              //System.out.println("concept links to attended object");
              // increase concept activation
              concepti.act = concepti.act + actinc;
              //also increase act to object
              activateObject(perc, actinc);
           } // end if concept links to attended object
        } // end for concepts
    } // end attend to object

    // must first create an object with location and size information
    // function activateObject will look for existing object and increment activation
    // if no existing object is found, activateObject will create new object
    // with initial activation of actinc
    public void objectAttention(double time, Percept item, double actinc) {
        activateObject(item, actinc);
    } // end objectAttention
    //---------
    // activates any existing object of specified type, regardless of location or size
    public void generalObjectAttention(double time, String type, double actinc) {
        //System.out.println("at general object attention with " + type);
        for (int i=0; i<masterObjectList.size(); i++) {
            Percept objecti = (Percept)masterObjectList.get(i);
            if (objecti.type==type) {
                //System.out.println("activation for " + objecti.type + " at " + objecti.x +
"," + objecti.y + " is " + objecti.act);
                activateObject(objecti, actinc);
                //System.out.println("activation becomes " + objecti.act);
            } // end if types match
        } // end for i master object list
    } // end generalObjectAttention
    //---------
    // activates any existing object at specified location
    // only affects objects, not concepts
    public void locationAttention(double time, int x, int y, double actinc) {
        for (int i=0; i<masterObjectList.size(); i++) {
            Percept objecti = (Percept)masterObjectList.get(i);
            if ((objecti.x==x)&(objecti.y==y)) {
                activateObject(objecti, actinc);
            } // end if x y match
        } // end for i master object list
    } // end location attention
    //---------
    public void testNeckerCubes() {
        initiateDomainAlpha();
        System.out.println("Necker Cube trial");
        System.out.println("without added attention");
        processAll(1.0, "a1.bmp");
        System.out.println(")))  time = 1.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(2.0, "a1.bmp");
        System.out.println(")))  time = 2.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(3.0, "a1.bmp");
        System.out.println(")))  time = 3.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(4.0, "a1.bmp");
        System.out.println(")))  time = 4.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(5.0, "a1.bmp");
        System.out.println("))) time = 5.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(6.0, "a1.bmp");
        System.out.println("))) time = 6.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(7.0, "a1.bmp");
        System.out.println("))) time = 7.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(8.0, "a1.bmp");
        System.out.println("))) time = 8.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(9.0, "a1.bmp");
        System.out.println("))) time = 9.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(10.0, "a1.bmp");
        System.out.println("))) time = 10.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(11.0, "a1.bmp");
        System.out.println("))) time = 11.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        processAll(12.0, "a1.bmp");
        System.out.println("))) time = 12.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();
        System.out.println("---------------");
        System.out.println("without added attention");
        printMasterObjectList();
        printMasterConceptList();
        printActiveConcepts();
        System.out.println("end of Necker cube trial");
        System.out.println("---------------");
    } // end testNeckerCubes

    public void testNeckerCubesWithAttention() {
        initiateDomainAlpha();
        System.out.println("Necker Cube with attention trial");
        System.out.println("conceptTime is " + conceptTime);
        System.out.println("old time is " + oldTime);
        System.out.println("active object list is " + activeObjects);
        System.out.println("---------------------");
        System.out.println("with general attention to objects of type corner ne 3d");

        System.out.println("))) time = 0.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();

        processAll(1.0, "a1.bmp");
        System.out.println("))) time = 1.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();

        processAll(2.0, "a1.bmp");
        System.out.println("))) time = 2.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();

        processAll(3.0, "a1.bmp");
        System.out.println("))) time = 3.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();

        processAll(4.0, "a1.bmp");
        System.out.println("))) time = 4.0:  active concepts " + activeConcepts);
        printMasterConceptList();
        printActiveConcepts();

        // attention to object can only affect concepts after initial concept recognition
        // concept processing happens at conceptDelta intervals; conceptDelta = 4.0
        // attention after time=4.0 will take affect when concepts are processed again at
time=8.0
        // (immediately affects act levels on master concept list, but does not yet affect
active concept list)
        System.out.println("-------------");
        // first create object; 3d corners are constructed on the basis of dotted-corner
features
        FeatureDetector feature1 = new FeatureDetector(cnedotsw, 7, 11, "cornerne3d");
        CornerNE3D corner1 = new CornerNE3D(feature1);
        // then add attention to object, and to related concepts
        attendToObject(4.1, corner1, 0.5);
        System.out.println("-------------");

        processAll(5.0, "a1.bmp");
        System.out.println("))) time = 5.0:  active concepts " + activeConcepts);
```

```
printMasterConceptList();
printActiveConcepts();

processAll(6.0, "a1.bmp");
System.out.println("))) time = 6.0:  active concepts " + activeConcepts);
printMasterConceptList();
printActiveConcepts();

processAll(7.0, "a1.bmp");
System.out.println("))) time = 7.0:  active concepts " + activeConcepts);
printMasterConceptList();
printActiveConcepts();

processAll(8.0, "a1.bmp");
System.out.println("))) time = 8.0:  active concepts " + activeConcepts);
printMasterConceptList();
printActiveConcepts();

System.out.println("-------------");
// first create object
FeatureDetector feature2 = new FeatureDetector(cswdotne, 11, 7, "cornersw3d");
CornerSW3D corner2 = new CornerSW3D(feature2);
// then attend to object
attendToObject(8.1, corner2, 0.5);
System.out.println("-------------");

processAll(9.0, "a1.bmp");
System.out.println("))) time = 9.0:  active concepts " + activeConcepts);
printMasterConceptList();
printActiveConcepts();

processAll(10.0, "a1.bmp");
System.out.println("))) time = 10.0:  active concepts " + activeConcepts);
printMasterConceptList();
printActiveConcepts();

processAll(11.0, "a1.bmp");
System.out.println("))) time = 11.0:  active concepts " + activeConcepts);
printMasterConceptList();
printActiveConcepts();

processAll(12.0, "a1.bmp");
System.out.println("))) time = 12.0:  active concepts " + activeConcepts);
printMasterConceptList();
printActiveConcepts();
```

```
        System.out.println("---------------");
        System.out.println("with general attention to objects of type corner ne 3d");
        printMasterObjectList();
        printMasterConceptList();
        printActiveConcepts();
    } // end testNeckerCubesWithAttention

    public void testSteadyInput(String inputFile) {
        System.out.println("================");
        initiateDomainAlpha();
        System.out.println("testing repeated input of same bitmap");
        System.out.println("with bitmap " + inputFile);
        processAll(1.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(2.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(3.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(4.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
    } // end testSteadyInput

    public void testSteadyInputLong(String inputFile) {
        System.out.println("================");
        initiateDomainAlpha();
        System.out.println("starting regularization of input example");
        processAll(1.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(2.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(3.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(4.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();

        processAll(5.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(6.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(7.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(8.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();

        processAll(9.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(10.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(11.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
        processAll(12.0, inputFile);
        printMasterObjectList();
        printMasterConceptList();
    } // end testSteadyInput

    public void testChangingInput(String file1, String file2, String file3) {
        System.out.println("================");
        initiateDomainAlpha();

        processAll(1.0, file1);
        printMasterObjectList();
        printMasterConceptList();
        processAll(2.0, file1);
        printMasterObjectList();
        printMasterConceptList();
        processAll(3.0, file1);
        printMasterObjectList();
        printMasterConceptList();
        processAll(4.0, file1);
        printMasterObjectList();
        printMasterConceptList();

        processAll(5.0, file2);
        printMasterObjectList();
        printMasterConceptList();
        processAll(6.0, file2);
        printMasterObjectList();
        printMasterConceptList();
        processAll(7.0, file2);
        printMasterObjectList();
        printMasterConceptList();
        processAll(8.0, file2);
        printMasterObjectList();
        printMasterConceptList();

        processAll(9.0, file3);
        printMasterObjectList();
        printMasterConceptList();
        processAll(10.0, file3);
        printMasterObjectList();
        printMasterConceptList();
        processAll(11.0, file3);
        printMasterObjectList();
        printMasterConceptList();
        processAll(12.0, file3);
        printMasterObjectList();
        printMasterConceptList();
    } // end testChangingInput
    //-----------------------------
    // constructor for Cycle
    public Cycle (String domainName) {
    }
    //----------------------------
    public static void main (String[] args) throws IOException{
        Cycle d = new Cycle("d3");

        //    changing input shows object persistence
        //       d.testChangingInput("e1.bmp","e2.bmp","e3.bmp");
        //    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        //    b1.bmp is three-sided square, shows regularization of input
        //       d.testSteadyInputLong("b1.bmp");
        //       System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        //       d.testNeckerCubes();
        //       System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        d.testNeckerCubesWithAttention();

        // comment out the above lines if you wish to suppress a trial
        // use "d.testSteadyInput" with various bitmap inputs
        // to create input, open Paint program, open b1.bmp file, select
view/zoom/custom/800%
```

```
        // make changes and save under a different name, as a monochrome bitmap file,
e.g.,  test1.bmp
        // files must be saved in same directory as this program
        //          d.testSteadyInput("test1.bmp");
        // will run the program on the new bitmap file
        // testSteadyInput runs for one cycle, 4 time units, long enough to trigger concept
recognition
        // for longer runs, use testSteadyInputLong
        //          d.testSteadyInputLong("test1.bmp");
        // to test changing input, use
        //          d.testChangingInput("test1.bmp", "test2.bmp", "test3.bmp");

   } // end main
} // end class Cycle
```

Appendix K. ApparentMotion.java

```
//ApparentMotion.java
//Katherine L. McCreight
// 9-15-04
//
// program detects apparent motion in sequences of dots

import java.io.*;
import java.util.StringTokenizer;
import java.util.*;
import java.lang.*;
import java.text.*;
import java.lang.Math.*;
import java.util.Arrays;

public class ApparentMotion {
   public LinkedList motionList = new LinkedList();
   public LinkedList segmentList = new LinkedList();
   public LinkedList pathList = new LinkedList();
   public double pathToSegmentIncrement;
   public LinkedList motionEastConceptList = new LinkedList();
   public LinkedList motionWestConceptList = new LinkedList();
   public LinkedList motionNorthConceptList = new LinkedList();
   public LinkedList motionSouthConceptList = new LinkedList();
   // need time-based store of features to record dots at t1, t2, t3
   public LinkedList[] timedFeatures = new LinkedList[3];
   public LinkedList t0Features = new LinkedList();
   public LinkedList t1Features = new LinkedList();
   public LinkedList t2Features = new LinkedList();
   public LinkedList motionEastObjectList = new LinkedList();
   public LinkedList motionWestObjectList = new LinkedList();
   public LinkedList motionNorthObjectList = new LinkedList();
   public LinkedList motionSouthObjectList = new LinkedList();
   public double standardDistance;
   public LinkedList activeFeatures = new LinkedList();
   public LinkedList newActiveFeatures = new LinkedList();
   public double actInitialValue;
   public double threshInitialValue;
   public double featureInputWeight;
   // implementation threshold screens out low-level input
   //featureImplementationThreshold: how much of bitgrid must match to initially detect
feature
   // featureThreshold: how much activation is required for feature to fire, be recognized
   public double featureImplementationThreshold;
```

```
  public double featureThreshold;
  // only need dot input
  int[] dot = {1,1,1,1,0,1,1,1,1};
  // make a matrix so each BitGrid is stored in the matrix location
  // corresponding to its input location
  public BitGrid[][] bitGridMatrix = new BitGrid[20][20];
  public FeatureDetector[][] fdList = new FeatureDetector[20][20];
  public FeatureDetector dot1$1 = new FeatureDetector(dot, 1, 1, "dot at 1,1");
  public FeatureDetector dot1$2 = new FeatureDetector(dot, 1, 2, "dot at 1,2");
  public FeatureDetector dot1$3 = new FeatureDetector(dot, 1, 3, "dot at 1,3");
  public FeatureDetector dot1$4 = new FeatureDetector(dot, 1, 4, "dot at 1,4");
  public FeatureDetector dot1$5 = new FeatureDetector(dot, 1, 5, "dot at 1,5");
....
//------------ sub classes ----------------

  class BitGrid  {
      int x;
      int y;
      int[] input = new int[9];

      // constructor
      BitGrid(int xloc, int yloc) {
        this.x = xloc;
        this.y = yloc;
      } // end constructor
  } // end class BitGrid
  // ------------
  class FeatureDetector  {
      String description;
      int x;
      int y;
      LinkedList neighbors = new LinkedList();
      LinkedList links = new LinkedList();
      double act;
      double baseThresh;
      double currentThresh;

      double bitAct;
      double objectAct;

      int[] inputCode = new int[9];

      double lastTime;
      double adjustedTime;

      public void getAct(double time) {
```

```
   this.act = this.bitAct + this.objectAct;
} // end method FeatureDetector.getAct


// features are stored on timed lists
// note time interval for this demonstration is fixed at one unit
public void fire(double time) {

   this.currentThresh = currentThresh - 0.1*currentThresh;
   this.lastTime = time;
   this.adjustedTime = this.lastTime;
   this.act = 0.0;
   activeFeatures.add(this);
   newActiveFeatures.add(this);

   if (time==0.0) {
         t0Features.add(this);
   } else if (time==1.0) {
         t1Features.add(this);
   } else if (time==2.0) {
         t2Features.add(this);
   } // end if else;
} // end method FeatureDetector.fire

// constructor
FeatureDetector(int[] v, int x, int y, String d) {
   this.x = x;
   this.y = y;
   this.inputCode = v;
   this.act = actInitialValue;
   this.baseThresh = threshInitialValue;
   this.currentThresh = baseThresh;
   this.description = d;
} // end constructor

FeatureDetector(int[] v, int x, int y) {
   this.x = x;
   this.y = y;
   this.inputCode = v;
   this.act = actInitialValue;
   this.baseThresh = threshInitialValue;
   this.currentThresh = baseThresh;
} // end constructor

FeatureDetector(int[] v) {
   inputCode = v;
```

```
         act = actInitialValue;
         baseThresh = threshInitialValue;
         currentThresh = baseThresh;
      } // end constructor

      FeatureDetector() {
         act = actInitialValue;
         baseThresh = threshInitialValue;
         currentThresh = baseThresh;
      } // end constructor

} // end class FeatureDetector
// -------
// motion is a feature-level element, with direction north, south, east, or west
// motion class is not relevant in deriving the apparent motion results
class Motion {
      String direction;
      double act;
      Motion (String d) {
         direction = d;
         act = 0.0;
      } // end constructor
} // end class motion

// ------
// motion segment has a direction (n, s, e, w), a start point (x1,y1), and an endpt (x2,y2)
class MotionSegment {
      String direction;
      int x1;
      int y1;
      int x2;
      int y2;
      double act;

      MotionSegment(String d, FeatureDetector a, FeatureDetector b) {
         direction = d;
         x1 = a.x;
         y1 = a.y;
         x2 = b.x;
         y2 = b.y;
      } // end constructor

} // end class motion segment

// ------
// path has a corridor (x=2, or y=7, e.g.) and a direction (n, s, e, w)
```

```
class Path {
        int corridor;
        String direction;
        double act;

        Path(int loc, String d) {
            corridor = loc;
            direction = d;
        } // end constructor

} // end class path

//----------
// methods

// feature activation comes from both bitGrid input and from active objects
// bitGrid info stored in bitGridMatrix
// note: must screen for derived features in off-screen locations
public void detectFeatures(double time) {
        //System.out.println("at detectFeatures at time " + time);
        bitGridActivation(time);
        getActiveFeatures(time);
} // end detectFeatures

// look at all feature lists, check xloc yloc to match bitgrid xloc yloc
// then, if location matches, check all feature detectors in that list
// shield null entries: note that array length is total length of openings, not length of
entries
public void bitGridActivation(double time) {
        ////System.out.println("at bitGridActivation");
        for(int x=0; x<20; x++) {
            for(int y=0; y<20; y++) {
                    BitGrid bgxy = bitGridMatrix[x][y];
                    if(!(bgxy==null)) {
                        FeatureDetector featureDetectorf = fdList[x][y];
                        if (!(featureDetectorf==null)) {
                                int partialActivation = compareVectors(bgxy.input,
featureDetectorf.inputCode);
                                double proportionalActivation = partialActivation / 9.0;

                                if (proportionalActivation>featureImplementationThreshold) {

                                    featureDetectorf.act = featureDetectorf.act +
featureInputWeight*proportionalActivation;
                                } // end if over threshold
                        } // end if not null feature
            } // end if not null bitgrid
        } // end for y
      } // end for x
} // end bitGridActivation

//-----------
// method fire puts features on global timedFeatures list
// must clear fdList before processing each set of timed input
public LinkedList getActiveFeatures(double time) {
      // first, reset activeFeatures list
      // method FeatureDetector.fire will add feature detectors to the list, activeFeatures
      activeFeatures = new LinkedList();
      for(int x=0; x<20; x++) {
         for(int y=0; y<20; y++) {
            FeatureDetector fff = fdList[x][y];
            if (fff!=null) {
               if (fff.act>featureThreshold) {
                     fff.fire(time);
               } // end if over threshold
            } // end if not null
         } // end for y
      } // end for x
      return activeFeatures;
} // end getActiveFeatures

//------
// if already on list, increment activation
// otherwise, add to list and initiate activation
public void activateMotion(Motion m, double newact) {
      boolean newMotion = true;
      for (int i=0; i<motionList.size(); i++) {
         Motion mi = (Motion)motionList.get(i);
         if (m.direction==mi.direction) {

            newMotion=false;
            mi.act = mi.act + newact;
         } // end if matching
      } // end for i

      if (newMotion) {
         m.act = m.act + newact;
         motionList.add(m);
      } // end if new
} // end activateMotion
//-----
// check for existing segments
```

```
   public void activateMotionSegment(MotionSegment m, double newact) {
         //System.out.println("at activateMotionSegment with " + m.direction);
         //System.out.println("normalized distance is bit distance / 4, weighted act is act /
distance");
         boolean newMotion = true;
         double weightedAct = 0.0;
         double bitdistance = 0.0;
         double distance = 0.0;
         if ((m.direction=="n")|(m.direction=="s")) {
            bitdistance = Math.abs(m.y2-m.y1);
         } else {
            bitdistance = Math.abs(m.x2-m.x1);
         }  // end if else ns, ew
         distance = bitdistance / standardDistance;
         weightedAct = newact / distance;
         //System.out.println("act " + newact + ", bit distance " + bitdistance + "
normalized distance "+ distance + ", weighted act " + weightedAct);
         for (int i=0; i<segmentList.size(); i++) {
            MotionSegment mi = (MotionSegment)segmentList.get(i);
            if ((m.direction==mi.direction)&(m.x1==mi.x1)&(m.y1==mi.y1)) {
                  //System.out.println("found match");
                  newMotion=false;
                  mi.act = mi.act + weightedAct;
            } // end if matching
         } // end for i

         if (newMotion) {
            //System.out.println("adding new motion to list");
            m.act = m.act + weightedAct;
            segmentList.add(m);
         } // end if new

   } // end activateMotionSegment

   //-------
   // for each pair of endpoints, across a time interval, create a motion element
   public void detectMotion(double time) {
         LinkedList then = new LinkedList();
         LinkedList now = new LinkedList();
         if (time==0.0) {
         } else if (time==1.0) {
            then = t0Features;
            now = t1Features;
         } else if (time==2.0) {
            then = t1Features;
            now = t2Features;
```

```
        } // end if else time
        for (int i=0; i<then.size(); i++) {
          FeatureDetector theni = (FeatureDetector)then.get(i);
          for (int j=0; j<now.size(); j++) {
                FeatureDetector nowj = (FeatureDetector)now.get(j);
                if ((theni.x==nowj.x)&(theni.y<nowj.y)) {
                  //motion south
                  Motion ms = new Motion("s");
                  activateMotion(ms, 0.5);
                } else if ((theni.x==nowj.x)&(theni.y>nowj.y)) {
                  //motion north
                  Motion mn = new Motion("n");
                  activateMotion(mn, 0.5);
                } else if ((theni.y==nowj.y)&(theni.x<nowj.x)) {
                  //motion east
                  Motion me = new Motion("e");
                  activateMotion(me, 0.5);
                } else if ((theni.y==nowj.y)&(theni.x>nowj.x)) {
                  //motion west
                  Motion mw = new Motion("w");
                  activateMotion(mw, 0.5);
                } // end if
          } // end for j now
        } // end for i then
    } // end detectMotion
    //------

    // for each pair of endpoints, across a time interval, create a motion segment
    // then increment activation of segments which fall in the corridor of a previously
established path
    public void detectMotionSegments(double time) {
        LinkedList then = new LinkedList();
        LinkedList now = new LinkedList();
        if (time==0.0) {
        } else if (time==1.0) {
          then = t0Features;
          now = t1Features;
        } else if (time==2.0) {
          then = t1Features;
          now = t2Features;
        } // end if else time
        for (int i=0; i<then.size(); i++) {
          FeatureDetector theni = (FeatureDetector)then.get(i);
          for (int j=0; j<now.size(); j++) {
                FeatureDetector nowj = (FeatureDetector)now.get(j);
                if ((theni.x==nowj.x)&(theni.y<nowj.y)) {
```

```
                //motion south
                MotionSegment ms = new MotionSegment("s", theni, nowj);
                activateMotionSegment(ms, 0.5);
            } else if ((theni.x==nowj.x)&(theni.y>nowj.y)) {
                //motion north
                MotionSegment mn = new MotionSegment("n", theni, nowj);
                activateMotionSegment(mn, 0.5);
            } else if ((theni.y==nowj.y)&(theni.x<nowj.x)) {
                //motion east
                MotionSegment me = new MotionSegment("e", theni, nowj);
                activateMotionSegment(me, 0.5);
            } else if ((theni.y==nowj.y)&(theni.x>nowj.x)) {
                //motion west
                MotionSegment mw = new MotionSegment("w", theni, nowj);
                activateMotionSegment(mw, 0.5);
            } // end if
        } // end for j now
    } // end for i then
    printPathList(time);
    System.out.println("before top-down spreading activation");
    printSegmentList();
    // priming occurs here
    spreadActivationFromPathsToSegments(time);
    System.out.println("after top-down spreading activation");
    printSegmentList();
} // end detectMotion

//------
// for any segments starting from the same origin, consider relative activations
// higher activation segment fires, lower activation segment is inhibited
public void resolveSegmentConflicts(double time) {
    //System.out.println("at resolve segment conflicts");
    LinkedList copy = new LinkedList();
    for (int a=0; a<segmentList.size(); a++) {
        MotionSegment tempa = (MotionSegment)segmentList.get(a);
        copy.add(tempa);
    } // end for a copy list
    for (int i=0; i<copy.size(); i++) {
        MotionSegment tempi = (MotionSegment)copy.get(i);
        for (int j=0; j<copy.size(); j++) {
            // this clause ensures that we don't compare an entry to itself (i=j)
            // and that we don't check pairs ji that have already been checked as ij
            if(j>i) {
                MotionSegment tempj = (MotionSegment)copy.get(j);
                //if same origin, compare activation levels
                if((tempi.x1==tempj.x1)&(tempi.y1==tempj.y1)) {
                    System.out.println("same origin for " + tempi.direction + "," +
tempj.direction);
                    System.out.println(tempi.direction + ": " + tempi.act);
                    System.out.println(tempj.direction + ": " + tempj.act);
                    if (tempi.act>tempj.act) {
                      segmentList.remove(tempj);
                      System.out.println("removed " + tempj.direction + " from active
list");
                    } else if (tempi.act<tempj.act) {
                      segmentList.remove(tempi);
                      System.out.println("removed " + tempi.direction + " from active
list");
                    } else if (tempi.act==tempj.act) {
                      // when activations are equal, consider destinations
                      // if same destination as another segment, don't use it
                      for (int m=0; m<segmentList.size(); m++) {
                          MotionSegment msm =
(MotionSegment)segmentList.get(m);
                          if
((msm.direction!=tempi.direction)&(msm.x2==tempi.x2)&(msm.y2==tempi.y2)) {
                            System.out.println("tempi has same dest as another
segment being considered");
                            System.out.println("removing " + tempi.direction + "
because of " + msm.direction + " ending at " + msm.x2 + "," + msm.y2);
                            segmentList.remove(tempi);
                          } else if
((msm.direction!=tempj.direction)&(msm.x2==tempj.x2)&(msm.y2==tempj.y2)) {
                            System.out.println("tempj has same dest as another
segment being considered");
                            System.out.println("removing " + tempj.direction + "
because of " + msm.direction + " ending at " + msm.x2 + "," + msm.y2);
                            segmentList.remove(tempj);
                          } // end if same dest as tempi else same dest as tempj
                      } // end for m
                    }// end if else i>j, i<j, i=j
                } // end compare
              } // end if j > i
          } // end for j
        } // end for i
  } // end resolve segment conflicts

  // for each motion segment, create or strengthen a path in that direction
  // for n/s motion, path records x-location; for e/w motion, path records y-location
  //------
  public void detectPaths(double t) {
        System.out.println("at detect paths, with existing path list " );
```

```
    printPathList(t);
    for (int i=0; i<segmentList.size(); i++) {
      MotionSegment mi = (MotionSegment)segmentList.get(i);
      if ((mi.direction=="s")|(mi.direction=="n")){
          Path p = new Path(mi.x1, mi.direction);
          activatePath(p, 0.5);
      } // end if n or s
      if ((mi.direction=="e")|(mi.direction=="w")) {
          Path p = new Path(mi.y1, mi.direction);
          activatePath(p, 0.5);
      } // end if e or w
    } // end for i segments
    System.out.println("after detecting new paths, path list becomes " );
    printPathList(t);
} // end detectPaths
//---------
// if previous path was one bit away, treat it as a match, including copying activation
public void activatePath(Path p, double newact) {
    //System.out.println("at activatePath with " + p.direction);
    LinkedList removeList = new LinkedList();
    boolean newPath = true;
    for (int i=0; i<pathList.size(); i++) {
      Path pi = (Path)pathList.get(i);
      int diff = Math.abs(pi.corridor-p.corridor);
      if ((p.direction==pi.direction)&(p.corridor==pi.corridor)) {
          //System.out.println("found match");
          newPath=false;
          pi.act = pi.act + newact;
      } else if ((p.direction==pi.direction)&(diff<2)) {
          // this handles corridor problem with changing input
          // copy activation from nearby path; remove older path below
          //System.out.println("found near match");
          p.act = pi.act;
          //System.out.println("copying act " + p.act + " from " + pi.corridor);
          removeList.add(pi);
      } // end if nearby corridor
    } // end for i
    if (newPath) {
      //System.out.println("adding new path to list");
      p.act = p.act + newact;
      pathList.add(p);
    } // end if new
    for (int r=0; r<removeList.size(); r++) {
      Path pr = (Path)removeList.get(r);
      pathList.remove(pr);
    } // end for r remove paths
```

```
  } // end activate path
  //--------------------------------
  // paths provide priming for resolving conflicts among possible perceived motions
  public void spreadActivationFromPathsToSegments(double time) {
        for (int i=0; i<pathList.size(); i++) {
           Path pi = (Path)pathList.get(i);
           for (int j=0; j<segmentList.size(); j++) {
                 MotionSegment msj = (MotionSegment)segmentList.get(j);
                 // if motion segment has same direction and corridor as path, strengthen
motion segment

                 // first check north and south motion
                 if ((msj.direction=="n")|(msj.direction=="s")) {
                    int xdiff = Math.abs(msj.x1-pi.corridor);
                    // should be xdiff=0 for same corridor
                    // allowing xdiff=1 for problem with changing input
                    if ((msj.direction==pi.direction)&(xdiff<2)) {
                          System.out.println("path reinforces segment " + msj.direction + "
at x=" + msj.x1);
                          msj.act = msj.act + pathToSegmentIncrement;
                    } // end if same direction, same or near corridor

                 } else {
                    // if not north/south, motion must be east/west
                    int ydiff = Math.abs(msj.y1-pi.corridor);
                    // should be ydiff=0 for same corridor
                    // allowing ydiff=1 for problem with changing input
                    if ((msj.direction==pi.direction)&(ydiff<2)) {
                          System.out.println("path reinforces segment " + msj.direction + "
at y=" + msj.y1);
                          msj.act = msj.act + pathToSegmentIncrement;
                    } // end if same direction, same or near corridor
                 } // end if else ns ew

           } // end for j segments
        } // end for i paths
  } // end spreadActivationFromPathsToSegments

  //-----
  // decay methods

  // together, decayFeatures and clearFeatureLists represent feature-level decay
  public void decayFeatures(){
        for (int i=0; i<20; i++) {
           for (int j=0; j<20; j++) {
                 FeatureDetector f = fdList[i][j];
```

```
                if (f!=null) {
                   f.act = 0;
                } // end if not null
             } // end for i
          } // end for j
       } // end clearfdListEntries

       public void clearFeatureLists() {
          t0Features = new LinkedList();
          t1Features = new LinkedList();
          t2Features = new LinkedList();
       } // end decayFeatures

       public void decayMotions() {
          motionList = new LinkedList();
       } // end decay motions

       public void decaySegments() {
          segmentList = new LinkedList();
       } // end decay segments

       public void clearPaths() {
          pathList = new LinkedList();
       } // end clear paths

       public void decayPaths() {
          System.out.println("at decayPaths");
          printPathList(0.0);
          double oldact;
          for (int i=0; i<pathList.size(); i++) {
             Path pi = (Path)pathList.get(i);
             oldact = pi.act;
             if (pi.act > 0.2) {
                pi.act = pi.act - 0.2;
                System.out.println("path activation for " + pi.direction + " " + oldact + "
decays to " + pi.act);
             } else {
                pi.act = 0.0;
                System.out.println("path activation for " + pi.direction +  " " + oldact + "
decays to " + pi.act);
             } // end if else
          } // end for i paths

          // can't remove paths from list above, would throw off loop indexing
          // remove any paths with 0 activation
          LinkedList copy = new LinkedList();
```

```
        for (int j=0; j<pathList.size(); j++) {
           Path pj = (Path)pathList.get(j);
           if (pj.act!=0.0) {
                 copy.add(pj);
           } // end if 0 activation
        } // end for j paths

        pathList = copy;
        printPathList(0.0);
  } // end decay paths

  //----
  // recognition methods

  // methods to input Paint files and extract bitmaps
  public LinkedList bytesEcho(String infile) {
        //System.out.println("at bytesEcho with file name " + infile);
        LinkedList wordBytes = new LinkedList();
        try {
           FileInputStream file = new FileInputStream(infile);
           BufferedInputStream buff = new BufferedInputStream(file);
           boolean eof = false;
           while (!eof) {
                 int input = buff.read();
                 ///System.out.println("read from input " + input);
                    Integer bigInput = new Integer(input);
                    wordBytes.add(bigInput);
                    if (input == -1) {
                          //System.out.println("found eof");
                          eof = true;
                    } // end if eof
           } // end while
           buff.close();
        } catch (IOException e) {
        } // end catch
        //System.out.println("read " + wordBytes);
        return wordBytes;
  } // end bytesEcho

  // get height and width information
  // height is 23rd character
  int getHeight (LinkedList input) {
        //System.out.println("at getHeight with " + input);
        int h = 0;
        Integer bigh = new Integer(0);
        bigh = (Integer)input.get(22);
        h = bigh.intValue();
        //System.out.println("h="+h);
        return h;
} // end getHeight

 // width is 19th character
int getWidth (LinkedList input) {
        int w = 0;
        Integer bigw = new Integer(0);
        bigw = (Integer)input.get(18);
        w = bigw.intValue();
        //System.out.println("w="+w);
        return w;
} // end get Width

// extract bitmap from bmp file
LinkedList bitTruncate(LinkedList input) {
        LinkedList output = new LinkedList();
        boolean bits = false;
        int[] whitecode = {255, 255, 255, 0};
        int[] temp = {-1, -1, -1, -1};
        // truncate everything up to and including 255 255 255 0
        for (int i=0; i< input.size(); i++) {
            Integer bigInt =(Integer)input.get(i);
            int in =bigInt.intValue();
            if (bits & (in!=-1)) {
                output.add(bigInt);
            } else {
                temp[0]=temp[1];
                temp[1] = temp[2];
                temp[2]=temp[3];
                temp[3] = in;
                if (temp[0]==whitecode[0] &
              temp[1]==whitecode[1] &
              temp[2]==whitecode[2] &
              temp[3]==whitecode[3]) {
                    bits = true;
                } // end if whitecode
            } // end not bits yet
        } // end for i
        return output;
} // end bitTruncate

public LinkedList binaryBits(LinkedList input) {
        LinkedList newbits = new LinkedList();
        // convert input digit into 8-bit binary code
        for (int i=0; i<input.size(); i++) {
            LinkedList bin = new LinkedList();
            Integer inputi = (Integer)input.get(i);
            int littleinputi = inputi.intValue();
            String binint = new String();
            int[] tempbits = {};
            Integer zero = new Integer(0);
            binint = Integer.toBinaryString(littleinputi);
            // convert to individual chars, to Strings, to ints
            // need to pad left: done in stringToBits
            tempbits = stringToBits(binint);
            //newbits.addAll(tempbits);
            for (int n=0; n<tempbits.length; n++) {
                int tempbitsn = tempbits[n];
                Integer bigtempbitsn = new Integer(tempbitsn);
                newbits.add(bigtempbitsn);
            } // end for n
        } // end for i
        return newbits;
    } // end binaryBits

    public int[] stringToBits (String bitstring) {

        //System.out.println("--at stringtobits with " + bitstring + " --");
        Integer zero = new Integer(0);
        int max = bitstring.length();
        int[] bits = new int[max];
        int[] padbits = new int[8];
        for (int i=0; i<max; i++) {
            char temp = bitstring.charAt(i);
            String temps = temp + "";
            int tempint = Integer.parseInt(temps);
            bits[i] = tempint;
        } // end for i
        int diff = 8-bits.length;
        // pad left of padbits if necessary to make 8 digits
        // if already 8 digits, diff = 0, d<diff will be false
        for (int d=0; d<diff; d++) {
            //padbits.addFirst(zero);
            padbits[d]=0;
        } // end for d
        // enter digits from bits into padbits, to the right of padded zeroes
        for (int e=0; e<max; e++) {
            int f = diff + e;
            padbits[f] = bits[e];
        } // end for e
        return padbits;
    } // end stringToBits

    // extract bits returns a list of rows, representing the original picture in bits
    // white = 1, black =0
    public LinkedList extractBits(LinkedList input) {
        //System.out.println("at extractBits with " + input);
        LinkedList output = new LinkedList();
        LinkedList bitoutput = new LinkedList();
        LinkedList rows = new LinkedList();
        // first get height and width information
        int h = getHeight(input);
        int w = getWidth(input);
        // then truncate header
        output = bitTruncate(input);
        //System.out.println("after calling bitTruncate: " + output);
        // then convert to binary
        bitoutput = binaryBits(output);
        int wide = 32;
        // then assemble based on height and width
        // each padded row = 32 bits
        // get a row and input only first w entries
        int rownumber = bitoutput.size()/wide;
        //System.out.println(rownumber + " rows");
        for (int dummy=0; dummy<rownumber; dummy++) {
          LinkedList temprow = new LinkedList();
          for (int m=0; m<wide; m++) {
                Integer bitoutputm = (Integer)bitoutput.get(m);
                temprow.add(bitoutputm);
          } // end for m
          // now remove 32 from bitoutput
          for (int p=0; p<wide; p++) {
                bitoutput.removeFirst();
          }  // end for p remove 32
          int wdiff = wide - w;
          for (int r=0; r<wdiff; r++) {
                temprow.removeLast();
          } // end for r remove right padding
          //System.out.println("temprow " + temprow);
          rows.addFirst(temprow);
        } // end for dummy
        //System.out.println("rows " + rows);
        return rows;
    } // end extractBits

    // bit grid matrix is 20 x 20
```

```
public void clearBitGridMatrix() {
  for (int i=0; i<20; i++) {
        for (int j=0; j<20; j++) {
              bitGridMatrix[i][j] = null;
        } // end for j
     } // end for i
} // end clearBitGridMatrix

// given input in row format from extractBits, compute 3x3 bit grids
// puts these into bitGridMatrix as a side-effect
// note:  avoid edges
public void bitGrids(LinkedList input) {
      clearBitGridMatrix();
      int h = getHeight(input);
      int w = getWidth(input);
      LinkedList rows = extractBits(input);
      for (int x=1; x<w-1; x++) {
         for(int y=1; y<h-1; y++) {
               //System.out.println("working on ("+ x + "," + y + ")");
               BitGrid bgxy = new BitGrid(x,y);
               LinkedList rowa = (LinkedList)rows.get(y-1);
               LinkedList rowb = (LinkedList)rows.get(y);
               LinkedList rowc = (LinkedList)rows.get(y+1);
               Integer bigInt0 = (Integer)rowa.get(x-1);
               int int0 = bigInt0.intValue();
               bgxy.input[0] = int0;

               Integer bigInt1 = (Integer)rowa.get(x);
               int int1 = bigInt1.intValue();
               bgxy.input[1] = int1;

               Integer bigInt2 = (Integer)rowa.get(x+1);
               int int2 = bigInt2.intValue();
               bgxy.input[2] = int2;

               Integer bigInt3 = (Integer)rowb.get(x-1);
               int int3 = bigInt3.intValue();
               bgxy.input[3] = int3;

               Integer bigInt4 = (Integer)rowb.get(x);
               int int4 = bigInt4.intValue();
               bgxy.input[4] = int4;

               Integer bigInt5 = (Integer)rowb.get(x+1);
               int int5 = bigInt5.intValue();
               bgxy.input[5] = int5;
```

```
                    Integer bigInt6 = (Integer)rowc.get(x-1);
                    int int6 = bigInt6.intValue();
                    bgxy.input[6] = int6;

                    Integer bigInt7 = (Integer)rowc.get(x);
                    int int7 = bigInt7.intValue();
                    bgxy.input[7] = int7;

                    Integer bigInt8 = (Integer)rowc.get(x+1);
                    int int8 = bigInt8.intValue();
                    bgxy.input[8] = int8;

                    bitGridMatrix[x][y] = bgxy;
                } // end for x
            }// end for y
        } // end bitGrids

//-------------------------------------------------------
    //print methods

    public void printMotionList(double time) {
        //System.out.println("motion list:");
        for (int i=0; i<motionList.size(); i++) {
            Motion mi = (Motion)motionList.get(i);
            System.out.println("  " + mi.direction + " act: " + mi.act);
        } // end for i
    } // end print motion list

    public void printSegmentList() {
        //System.out.println("segment list:");
        for (int i=0; i<segmentList.size(); i++) {
            MotionSegment msi = (MotionSegment)segmentList.get(i);
            System.out.println("  " + msi.direction + " act: " + msi.act + " (" + msi.x1 + ","
+ msi.y1 + ")-(" + msi.x2 + "," + msi.y2 + ")");
        } // end for i
    } // end print segment list

    public void printPathList(double time) {
        //System.out.println("path list:");
        for (int i=0; i<pathList.size(); i++) {
            Path pi = (Path)pathList.get(i);
            String coordinate = "";
            if ((pi.direction=="n")|(pi.direction=="s")){
                coordinate = "x";
            } else {
                coordinate = "y";
            } // end if else
            System.out.println("  " + pi.direction + " " + coordinate + "=" + pi.corridor);
        } // end for i
    } // end print path list

    //-------------------------------------------------
    // set domain variables
    public void initiateDomainAlpha() {
        standardDistance = 4.0;
        timedFeatures[0] = t0Features;
        timedFeatures[1] = t1Features;
        timedFeatures[2] = t2Features;
        pathToSegmentIncrement = 0.2;

        this.featureThreshold = 0.1;
        this.actInitialValue = 0.0;
        this.threshInitialValue = 1.0;

        // activation weights
        this.featureInputWeight = 1.0;

        // implementation thresholds
        // set to just under 9/9 for exact detection of dots
        this.featureImplementationThreshold = .99;

        // add feature lists to feature list matrix
        fdList[1][1] = dot1$1;
        fdList[1][2] = dot1$2;
        fdList[1][3] = dot1$3;
        fdList[1][4] = dot1$4;
        fdList[1][5] = dot1$5;
        fdList[1][6] = dot1$6;
        fdList[1][7] = dot1$7;
....
//utility methods

    // utility to check two binary inputs for matches
    // if both are 1, a&b is true
    // if both are 0, a|b is false
    public boolean bitmatch(int a, int b) {
        boolean match = false;
        if (((a&b)==1)|((a|b)==0)) {
            match = true;
        } // end if
        return match;
```

```
} // end bitmatch

// compare two binary vectors
// vectors must match in length
// uses utility, bitmatch
public int compareVectors(int[] inputa, int[] inputb) {
      int total = 0;
      for (int i=0; i<inputa.length; i++) {
         int a = inputa[i];
         int b = inputb[i];
         if ( bitmatch(a, b) ) {
               total = total + 1;
         } // end if bitmatch
      } // end for i
      //System.out.println("total number of matches is " + total);
      return total;
} // end compareVectors

//---------
// decay is not invoked automatically here, but is designated in the main program file
public void processAll(double time, String file) {
      LinkedList bits = bytesEcho(file);
      extractBits(bits);
      bitGrids(bits);
      detectFeatures(time);
      detectMotion(time);
      detectMotionSegments(time);
      resolveSegmentConflicts(time);
      detectPaths(time);
} // end processAll

//-----------------------------
// constructor for ApparentMotion
public ApparentMotion (String domainName) {
}
//----------------------------
public static void main (String[] args) throws IOException{
      ApparentMotion d = new ApparentMotion("d3");
      d.initiateDomainAlpha();

      System.out.println("processing with horizontal priming");
      d.processAll(0.0, "t0h.bmp");
      d.decayFeatures();
      d.decayMotions();
      d.decaySegments();
      d.processAll(1.0, "t1.bmp");
```

```
d.decayFeatures();
d.decayMotions();
d.decaySegments();
d.processAll(2.0, "t2.bmp");
System.out.println("motion perceived with horizontal priming:");
d.printSegmentList();

// now clear lists for new run
d.decayFeatures();
d.clearFeatureLists();
d.decayMotions();
d.decaySegments();
d.clearPaths();

System.out.println("---------------");
System.out.println("processing with vertical priming");
d.processAll(0.0, "t0v.bmp");
d.decayFeatures();
d.decayMotions();
d.decaySegments();
d.processAll(1.0, "t1.bmp");
d.decayFeatures();
d.decayMotions();
d.decaySegments();
d.processAll(2.0, "t2.bmp");
System.out.println("motion perceived with vertical priming:");
d.printSegmentList();

// now clear lists for new run
d.decayFeatures();
d.clearFeatureLists();
d.decayMotions();
d.decaySegments();
d.clearPaths();

System.out.println("---------------");
System.out.println("processing tall input");
d.processAll(1.0, "t1h3.bmp");
d.decayFeatures();
d.decayMotions();
d.decaySegments();
d.processAll(2.0, "t2h3.bmp");
System.out.println("motion perceived with tall input:");
d.printSegmentList();

// now clear lists for new run
```

```
d.decayFeatures();
d.clearFeatureLists();
d.decayMotions();
d.decaySegments();
d.clearPaths();

System.out.println("---------------");
System.out.println("processing wide input");
d.processAll(1.0, "t1h5.bmp");
d.decayFeatures();
d.decayMotions();
d.decaySegments();
d.processAll(2.0, "t2h5.bmp");
System.out.println("motion perceived with wide input:");
d.printSegmentList();

// now clear lists for new run
d.decayFeatures();
d.clearFeatureLists();
d.decayMotions();
d.decaySegments();
d.clearPaths();

System.out.println("---------------");
System.out.println("processing balanced input");
d.processAll(1.0, "t1h4.bmp");
d.decayFeatures();
d.decayMotions();
d.decaySegments();
d.processAll(2.0, "t2h4.bmp");
System.out.println("motion perceived for balanced input:");
d.printSegmentList();

// now clear lists for new run
d.decayFeatures();
d.clearFeatureLists();
d.decayMotions();
d.decaySegments();
d.clearPaths();

System.out.println("---------------");
System.out.println("processing increasing input");
System.out.println("trial 1:  distance n,s = 4, distance e,w = 2");
d.processAll(1.0, "t1h2.bmp");
d.decayFeatures();
d.decayMotions();
```

```
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h2.bmp");
        System.out.println("motion perceived on trial 1:  distance n,s = 4, distance e,w =
2");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 2:  distance n,s, = 4, distance e,w = 3");
        d.processAll(1.0, "t1h3.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h3.bmp");
        System.out.println("motion perceived on trial 2:  distance n,s, = 4, distance e,w =
3");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 3:  distance n,s, = 4, distance e,w = 4");
        d.processAll(1.0, "t1h4.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h4.bmp");
        System.out.println("motion perceived on trial 3:  distance n,s, = 4, distance e,w =
4");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 4:  distance n,s, = 4, distance e,w = 5");
        d.processAll(1.0, "t1h5.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h5.bmp");
        System.out.println("motion perceived on trial 4:  distance n,s, = 4, distance e,w =
5");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 5:  distance n,s, = 4, distance e,w = 6");
        d.processAll(1.0, "t1h6.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h6.bmp");
        System.out.println("motion perceived on trial 5:  distance n,s, = 4, distance e,w =
6");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 6:  distance n,s, = 4, distance e,w = 7");
        d.processAll(1.0, "t1h7.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h7.bmp");
        System.out.println("motion perceived on trial 6:  distance n,s, = 4, distance e,w =
7");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 7:  distance n,s, = 4, distance e,w = 8");
        d.processAll(1.0, "t1h8.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h8.bmp");
        System.out.println("motion perceived on trial 7:  distance n,s, = 4, distance e,w =
8");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();


        // now clear lists for new run
        d.decayFeatures();
        d.clearFeatureLists();
        d.decayMotions();
        d.decaySegments();
        d.clearPaths();

        System.out.println("---------------");
        System.out.println("processing decreasing input");

        System.out.println("trial 1:  distance n,s, = 4, distance e,w = 8");
        d.processAll(1.0, "t1h8.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h8.bmp");
        System.out.println("motion perceived on trial 1:  distance n,s, = 4, distance e,w =
8");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 2:  distance n,s, = 4, distance e,w = 7");
        d.processAll(1.0, "t1h7.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h7.bmp");
        System.out.println("motion perceived on trial 2:  distance n,s, = 4, distance e,w =
7");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 3:  distance n,s, = 4, distance e,w = 6");
        d.processAll(1.0, "t1h6.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h6.bmp");
        System.out.println("motion perceived on trial 3:  distance n,s, = 4, distance e,w =
6");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 4:  distance n,s, = 4, distance e,w = 5");
        d.processAll(1.0, "t1h5.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h5.bmp");
        System.out.println("motion perceived on trial 4:  distance n,s, = 4, distance e,w =
5");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 5:  distance n,s, = 4, distance e,w = 4");
        d.processAll(1.0, "t1h4.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h4.bmp");
        System.out.println("motion perceived on trial 5:  distance n,s, = 4, distance e,w =
4");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 6:  distance n,s, = 4, distance e,w = 3");
        d.processAll(1.0, "t1h3.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h3.bmp");
        System.out.println("motion perceived on trial 6:  distance n,s, = 4, distance e,w =
3");
        d.printSegmentList();
        System.out.println("- - - - - - - - - - ");
        d.clearFeatureLists();
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        System.out.println("trial 7:  distance n,s, = 4, distance e,w = 2");
        d.processAll(1.0, "t1h2.bmp");
        d.decayFeatures();
        d.decayMotions();
        d.decaySegments();
        d.decayPaths();
        d.processAll(2.0, "t2h2.bmp");
        System.out.println("motion perceived on trial 7:  distance n,s, = 4, distance e,w =
2");
        d.printSegmentList();

  } // end main
} // end class ApparentMotion
```

Appendix L. Domain.java

```
at domain constructor
at initiateDomainAlpha
line is b,c
line is b,c
line is b,c
line is l,m,n,o,p
line is l,m,n
line is o,p
output list is [[b, c], [b, c], [b, c], [l, m, n, o, p], [l, m, n], [o,
p]]
getData is [[b, c], [b, c], [b, c], [l, m, n, o, p], [l, m, n], [o, p]]
time 1.0
inputData is
[b, c]
input is b
input b matches node for b
activation becomes 0.5
input is c
input c matches node for c
activation becomes 0.5
targets of spreading []

short list []

after adding new firing nodes, []

removing duplicates []

time 2.0
inputData is
[b, c]
input is b
input b matches node for b
activation becomes 1.0
input is c
input c matches node for c
activation becomes 1.0
** saw b
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 2.0
threshold is now 0.9
** saw c
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 2.0
threshold is now 0.9
spreading from b to a
activation of a is now 0.25
targets of spreading [Domain$Node@360be0]
 a
short list [Domain$Node@360be0]
```

```
 a
after adding new firing nodes, [Domain$Node@45a877,
Domain$Node@1372a1a]
 b c
removing duplicates [Domain$Node@45a877, Domain$Node@1372a1a]
 b c
creating links between co-firing nodes

 b c
creating link from b to c
time is 2.0
nodes firing at time 2.0
 [Domain$Node@45a877, Domain$Node@1372a1a]
 b c
using b as representative node
checking for gravitational effects on node b
old domain state []

new domain state [Domain$Collocation@f6a746]
 2.0
this node fires at time 2.0
in domain state, there is a collocation at time 2.0
distance from current node is 0.0
no positive collocation, so no positive gravitation
no negative collocation, so no negative gravitation
total gravitation is 0.0

time 3.0
inputData is
[b, c]
input is b
input b matches node for b
activation becomes 0.5
input is c
input c matches node for c
activation becomes 0.5
targets of spreading []

short list []

after adding new firing nodes, []

removing duplicates []

time 4.0
inputData is
[l, m, n, o, p]
input is l
input l matches node for l
activation becomes 0.5
input is m
input m matches node for m
activation becomes 0.5
input is n
input n matches node for n
activation becomes 0.5
input is o
input o matches node for o
activation becomes 0.5
input is p
input p matches node for p
activation becomes 0.5
targets of spreading []

short list []

after adding new firing nodes, []

removing duplicates []

time 5.0
time reset to 0.0
old domain state [Domain$Collocation@f6a746]
 2.0
new domain state []

inputData is
[l, m, n]
input is l
input l matches node for l
activation becomes 1.0
input is m
input m matches node for m
activation becomes 1.0
input is n
input n matches node for n
activation becomes 1.0
node a may fire at will, since time has cycled to last firing time of
0.0
node d may fire at will, since time has cycled to last firing time of
0.0
node e may fire at will, since time has cycled to last firing time of
0.0
node l may fire at will, since time has cycled to last firing time of
0.0
node m may fire at will, since time has cycled to last firing time of
0.0
node n may fire at will, since time has cycled to last firing time of
0.0
node o may fire at will, since time has cycled to last firing time of
0.0
node p may fire at will, since time has cycled to last firing time of
0.0
** saw l
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 0.0
threshold is now 0.9
** saw m
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 0.0
```

```
threshold is now 0.9
** saw n
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 0.0
threshold is now 0.9
targets of spreading []

short list []

after adding new firing nodes, [Domain$Node@15ff48b,
Domain$Node@affc70, Domain$Node@1e63e3d]
 l m n
removing duplicates [Domain$Node@15ff48b, Domain$Node@affc70,
Domain$Node@1e63e3d]
 l m n
creating links between co-firing nodes

 l m n
creating link from l to m
creating link from l to n
creating link from m to n
time is 0.0
nodes firing at time 0.0
 [Domain$Node@15ff48b, Domain$Node@affc70, Domain$Node@1e63e3d]
 l m n
using l as representative node
checking for gravitational effects on node l
old domain state [Domain$Collocation@f6a746]
 2.0
new domain state [Domain$Collocation@1004901]
 0.0
this node fires at time 0.0
in domain state, there is a collocation at time 0.0
distance from current node is 0.0
in old domain state, there is a collocation at time 2.0
distance from current node is 2.0
using positive collocation firing at time 2.0
number of nodes in positive collocation 2.0
distance to positive collocation 2.0
nodeWeight 1.0
distanceWeight 1.0
positive gravitation 2.1
no negative collocation, so no negative gravitation
total gravitation is 2.1

slowed down node l to 0.2
slowed down node m to 0.2
slowed down node n to 0.2
time 1.0
inputData is
[o, p]
input is o
input o matches node for o
activation becomes 1.0
input is p
input p matches node for p
activation becomes 1.0
** saw o
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 1.0
threshold is now 0.9
** saw p
threshold was 1.0
activation was 1.0
last firing time was 0.0
current time is 1.0
threshold is now 0.9
targets of spreading []

short list []

after adding new firing nodes, [Domain$Node@1b90b39,
Domain$Node@18fe7c3]
 o p
removing duplicates [Domain$Node@1b90b39, Domain$Node@18fe7c3]
 o p
creating links between co-firing nodes

 o p
creating link from o to p
time is 1.0
nodes firing at time 1.0
 [Domain$Node@1b90b39, Domain$Node@18fe7c3]
 o p
using o as representative node
checking for gravitational effects on node o
old domain state [Domain$Collocation@f6a746]
 2.0
new domain state [Domain$Collocation@1004901,
Domain$Collocation@b8df17]
 0.0 1.0
this node fires at time 1.0
in domain state, there is a collocation at time 0.0
distance from current node is -1.0
in domain state, there is a collocation at time 1.0
distance from current node is 0.0
in old domain state, there is a collocation at time 2.0
distance from current node is 1.0
using positive collocation firing at time 2.0
number of nodes in positive collocation 2.0
distance to positive collocation 1.0
nodeWeight 1.0
distanceWeight 1.0
positive gravitation 3.1
using negative collocation firing at time 0.0
number of nodes in negative collocation 3.0
distance to negative collocation 1.0
nodeWeight 1.0
distanceWeight 1.0
negative gravitation -4.1
```

```
total gravitation is -0.9999999999999996

sped up node o to 0.8
sped up node p to 0.8
```

Appendix M. Installation Instructions

The programs need to be installed in a folder with the data files (mostly bitmaps). Here is a table showing the programs and their associated files.

Table 1. Programs and Data Files

Program	Data Files
Domain.java	data.txt
Cycle.java	a1.bmp, b1.bmp, e1.bmp, e2.bmp, e3.bmp
ApparentMotion.java	t1h2.bmp, t2h2.bmp, t1h3.bmp, t2h3.bmp, t1h4.bmp, t2h4.bmp, t1h5.bmp, t2h5.bmp, t1h6.bmp, t2h6.bmp, t1h7.bmp, t2h7.bmp, t1h8.bmp, t2h8.bmp, toh.bmp, tov.bmp

The programs Cycle.java and ApparentMotion.java use bitmap input. To view or edit the bitmaps, open them in Paint and select view/zoom/custom/800%. To turn grids on or off, select view/zoom/show grid. To edit a bitmap, save it in the program folder under a new name, with the extention .bmp (monochrome bitmap). To use the bitmap in the Cycle program, enter one of these function calls in the main section of the program:

```
d.testSteadyInput("name.bmp");
d.testSteadyInputLong("name.bmp");
d.testChangingInput("name1.bmp", "name2.bmp", "name3.bmp");
```

To use a bitmap in the ApparentMotion program, enter this function call in the main section of the program:

```
d.processAll("name.bmp");
```

Note that the decay functions are entered separately in the main section of this program; in order to use the decay functions, copy an entire section and replace the bitmap names with the name of the new bitmaps.

The program, Domain.java, uses the data file data1.txt. This file may be edited, or a new file may be created and called with the function call,

```
LinkedList getData = domain1.inputFile("name.txt");
```

Note that each line in a data file for Domain.java represents a new time interval.